AD-A067 579 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 9/2
ARITHMETIC CODES.(U)
DEC 78 T J PEREIRA

UNCLASSIFIED NL

1 OF 2
AD
A067579

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ARITHMETIC CODES

by

Tarcisio J. C. Pereira

December 1978

Thesis Advisor: Mitchell L. Cotton

Approved for public release; distribution unlimited

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Arithmetic Codes.		5. TYPE OF REPORT & PERIOD COVERED (9) Engineer's Thesis, December 1978
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Tarcisio J. C./Pereira		8. CONTRACT OR GRANT NUMBER(s) (12) 98p.
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE (11) Dec~~ember 1~~978
		13. NUMBER OF PAGES 99
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

CODES
Arithmetic codes
Fault tolerant computers

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis constitutes a tutorial study on the theory of arithmetic codes. Both non-separate and separate codes are discussed. Cyclic AN codes are presented and their analogy with cyclic parity check error correcting codes is emphasized. Also, a brief discussion of the implementation problem is carried out.

251 450 [illegible]

79 04 19 014

ARITHMETIC CODES

by

Tarcisio J. C. Pereira
Commander, Brazilian Navy

Submitted in partial fulfillment of the
requirements for the degree of

Electrical Engineer

from the

NAVAL POSTGRADUATE SCHOOL
December 1978

Author [illegible]

Approved by: Mitchell L. Cotton
Thesis Advisor

R. Panholzer
Second Reader

[illegible]
Chairman, Department of Electrical Engineering

William M. Tolles
Dean of Science and Engineering

ABSTRACT

This thesis constitutes a tutorial study on the theory of arithmetic codes. Both non-separate and separate codes are discussed. Cyclic AN codes are presented and their analogy with cyclic parity check error correcting codes is emphasized. Also, a brief discussion of the implementation problem is carried out.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. THESIS OBJECTIVES AND ORGANIZATION

A. FAULT TOLERANT COMPUTER SYSTEMS

In modern applications, exceptionally high reliability levels are often required from digital computers. Examples of such extremely severe requirements can be found in deep space missions and some military applications. In addition long non-critical jobs often require concurrent error detection. To achieve those high reliability levels one has available three basic strategies, two of which are also capable of providing concurrent error detection capability:

1. Use of highly reliable components in the manufacture of computers. This approach leads very rapidly to astronomical cost increases.

2. Use of redundancy—equipment redundancy or subsystems redundancy. In this scheme fault detection is achieved by comparison of outputs, and error correction through some majority voting criterion.

3. Use of more sophisticated redundancy schemes, such as encoding for error detection and/or correction.

The term fault tolerancy applies to those redundancy schemes designed to provide reliable operation in systems constructed with relatively unreliable components. The most widely accepted concept for fault tolerant computing system implies the existence of a built-in capability to preserve the continued correct execution of programs and input-output functions even in the presence of a certain set of operational faults. For our purposes an operational fault is a change of the value of one or more logical variables occurring as a consequence of a failure in the system's hardware.

The systematic study of fault tolerant systems was initiated in the early sixties, and has as one of its milestones the publication by William Pierce of his book, Failure Tolerant Computer Design. In this work, inspired by the results of Information Theory and Coding Theory, he attempted to apply the idea of redundancy to achieve high reliability in computer hardware.

The field of fault tolerance has grown significantly since and unifies various approaches to reliability assurance by means of testing, diagnosis and redundancy in machine organization and operation. It emerged in the late 1960's and reached maturity with the formation of the IEEE Computer Society Technical Committee on Fault-Tolerant Computing in 1969 and the subsequent holding of the First International Symposium on Fault-Tolerant Computing in 1971. This Symposium that has been held annually since, has became the major international forum for the discussion of current experience and new ideas in system design, redundancy techniques, system modeling and analysis, testing and diagnosis methods, and other related areas.

B. FORMULATION OF THE THESIS OBJECTIVE

The objective of this thesis is to study the theoretical fundamentals behind the fault tolerant design of arithmetic units. More specifically this study will be concerned with the so-called arithmetic error correcting codes.

Parity check error correcting codes have been extensively used in communications to ensure reliability in communications flowing through noisy channels. In a similar manner those codes are suitable for use in computer systems whenever transmission of information takes place between different subunits, to prevent error caused by failures in the circuitry involved. In addition, they may be used to provide the desired degree of fault tolerancy to memories of different types. Similarly, error correcting codes can be designed that will be conserved through arithmetic operations. An arithmetic code is a code such that, given a set of arithmetic operations, the result of those operations performed on operands represented in encoded form is the encoded result of the operation if performed in the original operands.

One should notice that in studying fault tolerance as achieved by the use of error correcting codes we often assume implicitly perfect, error free, decoding. It is interesting to comment that this assumption is, in the context of fault tolerant computer systems, stronger than when concerning error correcting codes in communications. In fact, the checking circuitry in fault tolerant computers is essentially the same kind of hardware used in the units to be checked, while in communications one can easily justify the belief that the decoder hardware is much more reliable than the noisy channel.

Even recognizing this fact we will, through all discussion in this thesis, assume perfect decoding, unless otherwise specified, since our main objective is to study the structure of the codes, as opposed to the evaluation of the resulting reliability improvement obtained when those codes are used in fault tolerant computers.

C. ORGANIZATION OF THE THESIS

The theory of the parity check error correcting codes used in communications has as one of its basic ideas the concept of distance between two code words. In very simple terms the distance between two code words, or in general, between two n-tuples $a_{n-1}, a_{n-2}, \ldots, a_i \ldots a_i a_o$ and $b_{n-1}, b_{n-2}, \ldots b_i \ldots b_i b_o$ is the number of positions such that $a_i \neq b_i$. If a code word w is transmitted and w_e is received, we say that k errors have occurred if the distance between w and w_e,

$$d(w,w_e) = k \qquad k = 0, 1, 2, \ldots n-1.$$

As mentioned before, the objective of our study--arithmetic codes--is codes that are preserved throughout a set of arithmetic operations, in particular addition, since this is the basic operation performed by computer arithmetic units.

It should be clear that the definition of distance and the resulting classification of errors as described above are not adequate as the basis for a theory in arithmetic codes. In effect, even if the adder cell is designed in such a way that a single hardware failure affects only one of the results or the carry bit, a single failure may result in the modification of the value of many code elements at the adder output, due to the propagation of the carry. A different concept of distance suitable for use in the theory of arithmetic codes is presented and developed in Chapter II.

We will discuss in this work two general classes of arithmetic codes: AN codes and separate codes. AN codes, that are discussed in Chapters III and IV, feature some inconvenient attributes as far as practical implementation is concerned. One such inconvenience is that the original information elements are not preserved in the encoded form. In addition, medium distance codes with large numbers of code words have not been found. Nevertheless, they are very important in many respects, particularly from a theoretical point of view. Some important classes of AN codes are in some respect analogous to some classes of algebraic codes. The basic theory of AN codes is studied in Chapter IV. In Chapter V a special class of codes analog to the cyclic algebraic codes is introduced and discussed.

Separate codes are those which preserve in the encoded form the original representation of the encoded information, concatenated with a set of check symbols. Those codes which are more attractive to practical implementation are introduced and discussed in Chapter V, where their relation with associated AN codes with the same error correcting and detecting capability is explored. Even though this thesis is mostly concerned with the theory of arithmetic codes, a brief discussion on implementation is carried out in Chapter VI. Finally, some conclusive comments as well as some topics open for future research are briefly presented in Chapter VII.

II. ARITHMETIC DISTANCE

A. ERRORS IN BINARY ADDITION

It is very important in the design of any error correcting code to take as a preliminary step the identification of the most likely modes of failure in the coded system.

This identification will provide the starting point for the formulation of a convenient description of the error that can be present at the system's output.

In this section three questions will be examined in detail for a binary adder, a unit which performs addition on two integers represented as binary numbers. At this particular point we will not impose any restrictions on those integers other than the requirement of being non-negative. The adder, in this fashion, may be a potentially infinite device, since no limitation has been imposed with respect to the size of the operands.

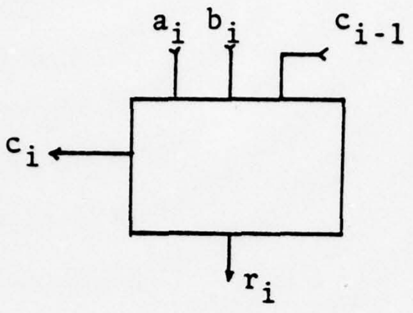
The radix-2 form of an integer I is the expression

$$I = a_0 + a_1 . 2^1 + a_2 . 2^2 + ... a_i . 2^i + ...$$

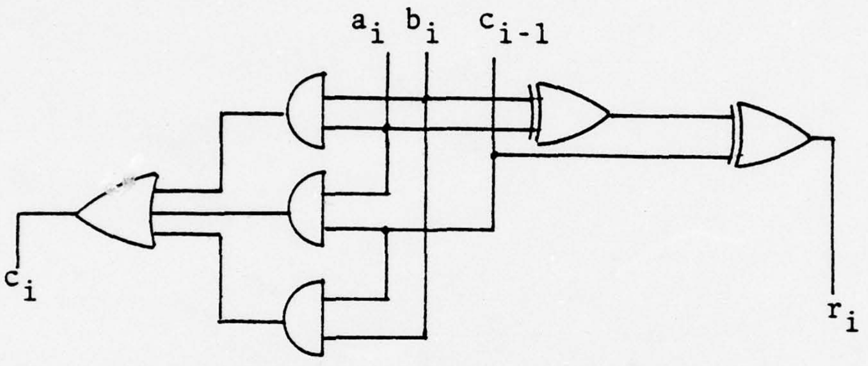
$$a_i \in \{0,1\} \qquad (2.1)$$

When $a_i = 0$ for all $i \geq n$ and $a_{n-1} \neq 0$, we usually write the radix-2 form as the n-tuple

$$a_{n-1}, a_{n-2}, \dots a_i \dots a_1, a_0$$



(a) block diagram



(b) logic diagram

a_i	b_i	c_{i-1}	r_i	c_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(c) truth table

Fig. 2.1. Full Adder. Single component failure will not cause C_i and r_i be in error simultaneously.

and we say that n places are required to represent I in radix-2 form.

The adder circuitry can be thought of as a sequence of elemental building blocks--full adders in integrated circuit terminology--that are associated with each binary position or power of 2. The elemental adder has as inputs the coefficients of the two operands, corresponding to a given position in their radix-2 forms and the carry propagating from the previous position. It generates at its output the respective coefficient of the radix-2 form of the result and the carry to the next binary position. An adder cell is represented in Figure 2.1.

For the situation illustrated in Figure 2.1 it seems natural to consider that a single error has occurred whenever an incorrect sum r_i or an incorrect carry c_i is generated. In consequence of this concept a single error in the adder output will correspond to the addition of $\pm 2^j$ to the correct result of its operation.

Suppose now that the actual result produced by a binary adder differs by E from the correct result. E can be represented as

$$E = \sum_{i=0}^{n-1} e_i 2^i \quad e_i \varepsilon \{-1, 0, +1\} \tag{2.2}$$

It is then natural to consider that the number of errors occurred--the arithmetic weight of E--as being the minimum number of e_i for which $e_i \neq 0$ in an expression such as (2.2). For example, E = 7

$$7 = 1 \times 2^2 + 1 \times 2 + 1 \times 2^0 = 1 \times 2^3 - 1 \times 2^0$$

$$w(7) = 2$$

B. ARITHMETIC WEIGHT

To formalize the concept introduced in the previous section, let us consider an error E, which may be positive, negative or zero. We can express E as

$$E = e_o + e_1 \; 2^1 + e_2 \; e^2 + \ldots e_i 2^i + \ldots$$

$$e_i \in \{-1, 0, 1\} \qquad (2.3)$$

Such an expression is called a modified binary form for E, and in general is not unique as illustrated in the example in Section II.A. It can be easily seen from (2.3) that if 2^i divides E then $e_o = e_i = e_{i-1} = 0$.

Definition 2.1

The arithmetic weight of the integer E, denoted W(E), is the least number of nonzero coefficients in a modified binary form for E. Besides the fact that an expression of the form given in (2.3) is not unique in general, it is possible that more than one of those expressions have the minimum number of nonzero terms.

For example, $3 = 1 \times 2^1 + 1 \times 2^0 = 1 \times 2^2 - 1 \times 2^0$

and thus, the weight of 3 is 2, but there are two different expressions that achieve the minimum of two nonzero terms. There is, however, a canonical form for every number which has the minimum number of terms. This form, introduced by

Reitwiesner, is unique and has proved of much importance in the theory of arithmetic coding. It is called the non-adjacent form (or NAF for short) if the coefficients e_i satisfy $e_i e_{i+1} = 0$ for $i = 0, 1, 2, \ldots$, in other words, if there are no two adjacent nonzero coefficients in its expression.

Theorem 2.1

For every number N there is a unique representation of the form

$$E = e_{n-1}\, 2^{n-1} + \ldots + e_0$$

in which $\overset{\ell}{e_i} = \underline{+}1$ or 0, and in which no two successive e_i are both nonzero. This representation has the minimum number of nonzero terms.

Proof

First, the existence of a representation of this type with a minimum number of nonzero terms will be demonstrated, and then the uniqueness will be demonstrated. Let

$$E = e_{n-1}\, 2^{n-1} + \ldots + e_0 \qquad (2.4)$$

If there exists two successive nonzero e_i, then let the pair with the smallest i be $e_{i+1}\, 2^{i+1} + e_i\, 2^i$. If $e_i = -e_{i+1}$, then $e_{i+1}\, 2^{i+1} + e_i\, 2^i = e_{i+1}\, 2^i$ and the resulting expression has fewer ones. If $e_i = e_{i+1}$, then

$$e_{i+1} 2^{i+1} + e_i 2^i = e_{i+1} 2^{i+2} - e_i 2^i.$$

If this substitution is made in Equation 2.4, the coefficient of 2^{i+1} will become zero. The term $e_{i+1} 2^{i+2}$ can be combined with the existing term $e_{i+2} 2^{i+2}$. If e_{i+2} is zero, the new expression has the same number of terms as the old. If $e_{i+2} = -e_{i+1}$, these terms cancel and there are fewer terms. If $e_{i+2} = e_{i+1}$, then combined they become

$$e_{i+2} 2^{i+2} + e_{i+1} 2^{i+2} = e_{i+1} 2^{i+3}$$

and the coefficient of 2^{i+2} becomes zero, while the $e_{i+1} 2^{i+3}$ must be combined with the term $e_{i+3} 2^{i+3}$. The "carry" may propagate to the highest-order term, but each time a carry occurs, the number of nonzero terms decreases. Thus, the resulting expression has the same general form given by Equation 2.4, and has no more nonzero terms than the original expression, and has no two successive nonzero terms of degree less than i+2.

If this process is repeated, an expression will result that has no two successive nonzero terms, and no more nonzero terms than the original expression. The process will certainly terminate because each step leaves at least two more terms in the low-order part which has no pair of successive nonzero terms, and there is at each step no more terms in all than the initial number, which was certainly finite.

If the original expression used had a minimum number of terms, the number of terms can decrease no more, and so the final expression must have exactly the same number of terms. Since every number certainly has an expression of the form in Equation 2.3 with a minimum number of terms, it follows that every number has an expression of that same type with a minimum number of terms and no two successive nonzero terms.

It remains to be shown that the expression is unique. First, note that in an expression of the type in Equation 2.3 with all coefficients ± 1 or 0, if the leading coefficient is +1, then the smallest value of E could occur if all other coefficients are -1, and in that case

$$E_{min} = 2^{n-1} - 2^{n-2} - \ldots -1 = 1$$

Similarly, if the leading coefficient is -1, the largest value E could have would be -1. *Thus, an expression can be* zero if and only if all coefficients are zero.

Now define

$$E_1 = a_{n-1} + \ldots + a_0$$

$$E_2 = b_{n-1}\, 2^{n-1} + \ldots + b_0$$

and suppose that no two successive coefficients in either expression are nonzero, and that for at least one i, $a_i \neq b_i$. Then

$$E_2 - E_1 = (b_{n-1} - a_{n-1})\, 2^{n-1} + (b_{n-2} - a_{n-2})\, 2^{n-2} + \ldots + (b_0 - a_0).$$

Then, if $a_i \neq b_i$, there are two cases; either $a_i = -b_i$ or one of them is zero. In the latter case, the expression for $E_2 - E_1$ has a nonzero term $(b_i - a_i)\ 2^i$, since a carry from the (i-1)th position is impossible. In the former case the term $2b_i\ 2^i = b_i\ 2^{i+1}$ results. But since neither the expression for E_1 or for E_2 has a pair of successive nonzero terms, $b_{i+1} = a_{i+1} = 0$, the term $b_i\ 2^{i+1}$ remains in the expression for $E_2 - E_1$. If all pairs of terms $a_i\ 2^i$ and $b_i\ 2^i$ are combined in this manner, the resulting expression has the form in Equation 2.3. Not all coefficients are 0, and therefore, $E_1 - E_2 \neq 0$, so that two distinct expressions give unequal numbers.

This proof shows how to express any number in the canonical form and thus determine the arithmetic weight. As an example, let us take 431 or, in binary notation 1 1 0 1 0 1 1 1 1.

$$\begin{aligned} 431 &= 2^8 + 2^7 + 0 + 2^5 + 0 + 2^3 + 2^2 + 2 + 1 \\ &= 2^8 + 2^7 + 0 + 2^5 + 2^4 + 0 + 0 + 0 - 1 \\ &= 2^8 + 2^7 + 2^6 + 0 - 2^4 + 0 + 0 + 0 - 1 \\ &= 2^9 + 0 + 0 - 2^6 + 0 - 2^4 + 0 + 0 + 0 - 1 \end{aligned}$$

and the arithmetic weight is 4, since there are 4 nonzero terms in the final expression.

The arithmetic weights of the first few numbers is as follows:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	2	1	2	2	2	1	2	2	3	2	3	2	2	1	2

18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
2	3	2	3	3	3	2	3	3	3	2	3	2	2	1

The following pattern was first observed by Chiang and Reed (1970). Starting from any number $2^i + 1$ up to $2^i + 2^{i-1}$, the weights are just one greater than the weights of 1 through 2^{i-1}. Starting from $2^i + 2^{i-1} + 1$ and going on to 2^{i+1}, the weights are the same as for the 2^{i-1} numbers $2^i + 2^{i-1} - 1$, $2^i + 2^{i-1} - 2$, ..., 2^i.

An alternative, and somewhat easier way of computing the NAF is provided by the following theorem, due to Chiang and Tsao-Wu:

<u>Theorem 2.2</u>

Let $b_{n-1}, \ldots, b_1, b_0$ and $a_{n-1}, \ldots, a_1, a_0$ be the radix-2 forms of the positive integers 3E and E respectively; then, the coefficients e_i defined by

$$e_i = b_{i-1} - a_{i+1} \quad \text{for } i = 0, 1, \cdots, n-2$$

$$e_i = 0 \qquad\qquad \text{for } i \geq n-1$$

are the coefficients in the NAF for E.

<u>Proof</u>

We note first that 3E and E are both even or both odd, so that $b_0 = a_0$. Hence,

$$\sum_{i=0}^{n-2} e_i 2^i = \sum_{i=0}^{n-2} (b_{i+1} - a_{i+1}) 2^i$$

$$= 1/2 \sum_{i=0}^{n-1} (b_i - a_i) 2^i$$

$$= 1/2\ (3E-E) = E$$

so that the e_i are certainly the coefficients in a modified binary form for E and it remains to show that this form is nonadjacent, i.e., that $e_i e_{i+1} = 0$ for all i. If $e_i = 0$, there is nothing to show. Suppose then that $e_i = -1$, i.e., that $b_{i+1} = 0$ and $a_{i+1} = 1$. We next note that we can obtain 3E by adding the radix-2 forms for E and 2E, the latter being just $a_{n-2}, \ldots a_0, 0$. Hence, the coefficients in the radix-2 form of 3E are given by

$$b_{i+1} = a_{i+1} + a_i + c_i \pmod 2$$

where c_i is the carry bit from the addition which gave b_i. But by hypothesis, $b_{i+1} = 0$ and $a_{i+1} = 1$, so that we must have

$$a_i + c_i = 1 \pmod 2.$$

Thus, two of the bits being added to give b_{i+1} must be "ones" so that the next carry $c_{i+1} = 1$. But this in turn implies

$$b_{i+2} = a_{i+2} + a_{i+1} + c_{i+1} \pmod 2$$
$$= a_{i+2}$$

so that $e_{i+1} = b_{i+2} - a_{i+2} = 0$ and hence $e_i\ e_{i-1} = 0$, as was to be shown. An entirely similar argument holds for $e_i = +1$. In simple terms Theorem 2.2 shows that the NAF of a positive integer can be determined by the bit by bit radix-2 subtraction of 3E and E. Clearly, if E is negative the NAF of E can be obtained by first evaluating the NAF of -E and then changing the sign of its coefficients.

Let's introduce the convenient notation that, in a NAF $e_{n-1}, e_{n-2} \ldots e_j \ldots e_0$ the symbol I in the jth position indicates the coefficient $e_j = -1$. The following example illustrates the procedure.

```
Ex          E = - 125        -E = 125
    3 x 125 = 375          101110111
          125                1111101
                           ---------
    NAF for 125            10000Ī01
    NAF for -125           Ī0000101
```

As an immediate consequence of Theorem 2.2 we have the following.

Corollary 2.1

The number of places required for the NAF of E (E a positive integer) is one less than the number of places required for the radix-2 form of 3E. In particular, the number of places required for the NAF of E is at most one greater than the number of places required for the radix-2 form of E.

C. ALGORITHM TO COMPUTE NAF AND ARITHMETIC WEIGHT

Theorem 2.2 suggests a very simple and convenient algorithm to compute the NAF at a given number, and consequently to determine its arithmetic weight, with the assistance of a digital computer.

Algorithm 2.1

Step (0). Let a_{n-1}, ... a_1, a_0, be the radix-2 form for a positive integer E such that $a_{n-1} = a_{n-2} = 0$. Set $i = 0$ and $c_0 = 0$.

Step (1). Calculate in order

$$b_{i+1} = a_{i+1} + a_i + c_i \pmod 2$$

$$c_{i+1} = \text{carry bit from the above addition}$$

$$e_i = b_{i+1} - a_{i+1}.$$

Step (2). If $i = n-2$, stop. Otherwise, increase i by 1 and return to step (1). When the calculaton has stopped, e_{n-2}, ..., e_i, e_0 is the desired NAF for E.

Very often we are interested in computing only the arithmetic weight of a number E, W(E). The following algorithm, due to Garcia, is directly inferable from algorithm 2.1 and permits the direct calculation of W(E) from the radix-2 form of E, without the computation of its NAF.

Algorithm 2.2

Step (1). Let $[a_{n-1}, \ldots, a_1, a_0]$ be the radix-2 form for a positive integer E such that $a_{n-1} = a_{n-2} = 0$. Set $i = 0$, $W = 0$, and $a_{-1} = 0$.

Step (2). If $i \geq n-2$, stop. Otherwise, increase i by 1 and go to step (1). When the calculation has stopped, $W = W(E)$.

It should be mentioned that Chiang and Reed have given an algorithm for calculating W(E) that does not require a radix-2 form for E as a starting point but works directly with integers. Also, Goto and Fukumura and Chien et al. have given algorithms for the computation of the NAF of E based upon nonrestoring binary division.

D. ARITHMETIC DISTANCE

In the theory of error correcting codes the concept of Hamming distance, the Hamming weight, $W(C_1 + C_2)$ where C_1 and C_2 are two code words, is shown to be a true metric of the code space.

In the theory of arithmetic codes we also recognize that the arithmetic weight satisfies the following properties of a norm.

$$W(E) = W(-E) \tag{2.5}$$

$$W(E) \geq 0 \text{ with equality if and only if } E = 0 \tag{2.6}$$

$$W(E + E') \leq W(E) + W(E') \tag{2.7}$$

Properties 2.5 and 2.6 follow directly from definition 2.1 for arithmetic weight. Property 2.7 is proved by noting that if the NAF's of E and E' are added digit-by-digit, then carries can flow only into the adjacent terms where the digit-by-digit sum is zero, hence E + E' can certainly be written as a modified binary form (which may not be an NAF) with at most as many nonzero terms as the sum of the number of nonzero terms in the two NAF's being added. But W(E + E') is the least number of terms in any modified binary form for E + E', so that 2.7 follows.

The norm properties (2.5) through (2.7) of arithmetic weight guarantee that a true metric for the set of integers can be defined as follows.

Definition 2.2

The arithmetic distance between the integers I_1 and I_2, denoted $D(I_1, I_2)$, is the arithmetic weight of their difference, i.e.,

$$D(I_1, I_2) = W(I_1 - I_2) \tag{2.8}$$

It is readily checked that for any integers I_1, I_2 and I_3, the following three properties which formally define a metric, are satisfied by arithmetic distance:

$$D(I_1, I_2) = D(I_2, I_1) \tag{2.9}$$

$$D(I_1, I_2) \geq 0 \text{ with equality if and only if } I_1 = I_2 \tag{2.10}$$

$$D(I_1, I_2) \leq D(I_1, I_3) + D(I_3, I_2) \tag{2.11}$$

The above properties which are called the symmetry, positive-definite, and triangle-inequality properties, follow directly from (2.5), (2.6) and (2.7) respectively.

As a consequence of (2.9) through (2.11), any set of integers with arithmetic distance taken as the measure of "distance" forms a true metric space. This fact plays an important role in the study of codes for the correction of errors in integer addition.

Finally, suppose that the correct result of some operation is the integer J but the actual result is $I = J + E$, where E is an error with $W(E) = i$. It follows from (2.8) that

$$D(I, J) = i$$

i.e., that an error of weight i in some result causes the erroneous result to be at arithmetic distance i from the true result.

E. ERROR IN COMPUTER ARITHMETIC

In previous sections we have concluded that arithmetic weight provides a good criteria to measure error in ordinary addition of integers. However, ordinary arithmetic is not used in digital computers. Those in fact perform module m arithmetic, with m usually either 2^n or 2^n-1. The corresponding arithmetics are respectively known as two's complement arithmetic and one's complement arithmetic. In Chapter III these arithmetics will be considered in greater detail, but at this point it seems useful to analyze the carry propagation in modulo addition. Our objective is to learn what happens when an error is made in the carry from the highest, left-most adding position to conclude under which conditions, if any, such an error will show up in the result as a single error, when measured by the arithmetic weight.

A suitable way of analyzing the carry propagation from the highest adding position, supposing that the adder has n adding cells (2^0, 2^1, ... 2^{n-1}), is to consider the result of the operation $2^n-1 + 2^n-1$ or $2(2^n-1)$.

Refer to Figure 2.2, where a n cell adder is represented.

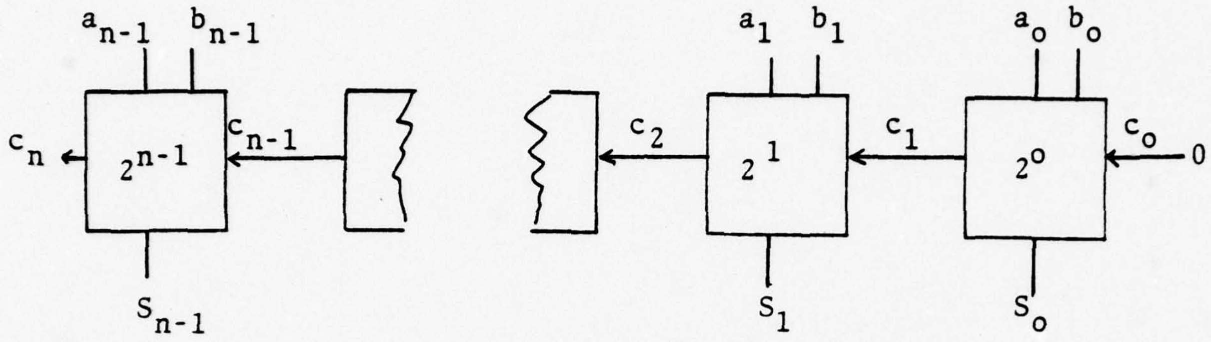


Fig. 2.2. Binary Adder Block Diagram.

The question we are addressing is how to connect C_n for $m = 2^n$, $m = 2^n-1$, and m other than 2^n or 2^n-1.

Let's proceed this analysis for the three cases above.

Case 1 - $m = 2^n$

```
  1 ........ 000
+ 1 ........ 000
1 00 ....... 000
```

For this case the carry is irrelevant as far as the arithmetic is concerned.

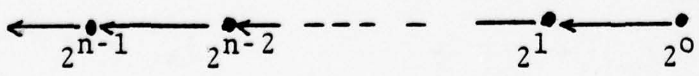


Fig. 2.3. Carry Propagation for $m = 2^n$.

Case 2. $m = 2^{n-1}$

```
    1 ........ 000
    1          000
    --------------
  1 0 ........000
-   1         111  -(2^n-1)
  ----------------
  0 0 ........001
```

The carry propagates to the least significant position only. An error in the carry will therefore show up as a unitary weight error.

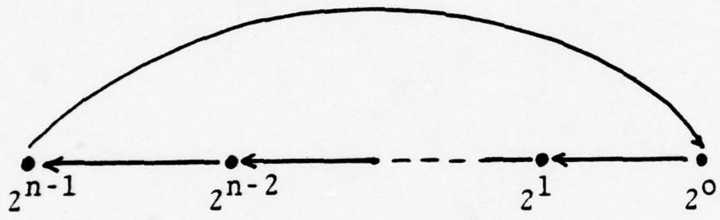


Fig. 2.4. Carry Propagation for $m = 2^n-1$.

For these two cases we see that the arithmetic weight of any error has the same significance as it has for the ordinary addition of integers, namely that t failures in the elemental adding units (each failure being an incorrect modulo-two sum or an incorrect carry bit) still creates an error E of arithmetic weight at most t in the sum. This follows from the fact that the carry bit generated by any elemental adding unit flows to at most one other unit. Thus, the arithmetic weight of an error is still an appropriate measure of the amount of logical manfunction in the adder just as it was for the ordinary addition of integers in radix-2 form. It should be pointed out of course that if, in an elemental adding unit, the logic which performs the modulo-two addition is shared with that which generates the carry bit, then a single logical malfunction can create an error of arithmetic weight two. This remark applies to the ordinary addition of integers as well as the modulo m addition for $m = 2^{n-1}$ or $m = 2^n$.

<u>Case 3</u>

Suppose now for illustration that m is for example 13. Four binary positions are required in the adder so

$$\begin{array}{rl} 1000 & \\ \underline{1000} & \\ 10000 & \\ -\underline{\ 1101} & (-13) \\ 0011 & \end{array}$$

It follows that an error in the carry from the left most adding unit results in an error of arithmetic weight W(E) = 2.

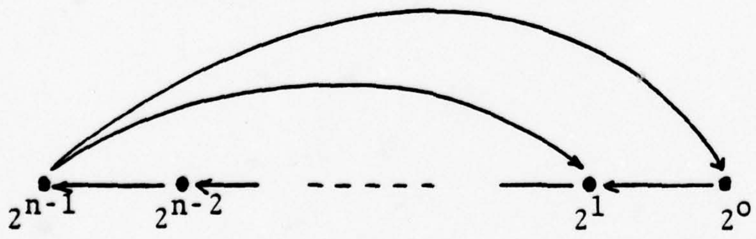


Fig. 2.5. Carry Propagation for m = 13.

III. BASIC THEORY OF AN CODES

An AN code generated by a positive integer A is the set of integers AN for $0 \leq N \leq B$. From the definition above, it is apparent that B is the number of integers in the code.

The integers belonging to an AN code will be referred to as the code points. So the code point AN_1 is to be understood as the encoded version of the information integer N_1.

Since

$$AN_1 + AN_2 = A(N_1 + N_2)$$

the ordinary sum of the code points N_1 and N_2 is the code point corresponding to the integer $N_1 + N_2$, provided

$$N_1 + N_2 \leq B \tag{3.1}$$

In this restricted sense AN codes are linear, and are sometimes called linear residue codes. The restriction imposed on the linearity of AN codes is that expressed in (3.1).

A. MINIMUM ARITHMETIC DISTANCE AND ERROR CORRECTION

Definition 3.1

The minimum arithmetic distance D_{min} of a AN code is the minimum of arithmetic distances between all pairs of distinct code points. The linearity of AN codes leads to the property given in the theorem below.

Theorem 3.1

The minimum arithmetic distance D_{min} of an AN code is equal to the minimum arithmetic weight W_{min} of the B-1 nonzero code points in the code

$$D_{min} = W_{min} = \min W(AN) \qquad 0 < N < B$$

Proof

If AN_1 and AN_2 are code points with $N_1 > N_2$, then

$$D(AN_1, AN_2) = W[A(N_1-N_2)] = W(AN_3)$$

where $0 < N_3 = N_1 - N_2 < B$, which shows that the distance between any pair of distinct code points is always the weight of some nonzero code point. Conversely, if AN_3 is any nonzero point, then

$$D(0, AN_3) = W(AN_3)$$

so that the weight of every nonzero code point is also the distance between some pair of distinct code points and therefore the theorem is proved.

The minimum arithmetic distance of an AN code determines its capability to correct and detect errors in its code points. We next establish this relation.

In the discussion to follow we assume that the correct result of some operation is some code point in the AN code and that any code point is a possible result. For example, the operation might be addition (without overflow) of code points. If the correct result is AN_1 but the actual result is $I = AN_1 + E$ we will say that an error E has occurred, and call E the actual error.

The decoder, which is the device which does the actual error correction or detection, can operate only on the actual result, I. The basic theorem which governs the error correcting and detecting capability of an AN code is as follows.

Theorem 3.2

For any $t \geq 0$ and any $s \geq 0$, an AN code can correct all errors of weight t or less in its code points and can detect all other errors of weight $t + s$ or less in its code points if and only if its minimum arithmetic distance satisfies

$$D_{min} > 2t + S$$

Proof

Suppose that $D_{min} > 2t + s$. We next show that the decoder which decodes the result I into the nearest code point AN_1 if there is a code point AN_1 with $D(I, AN_1) \leq t$, and announces a detected error if $t < D(I, AN_1) \leq t + s$ performs the claimed correction and detection. For, suppose $I = AN + E$, where AN is the correct result and $W(E) \leq t$, then, $D(I, AN) \leq t$. But for any other code point AN', we have by the triangle inequality that $D(I, AN') \geq D(AN, AN') - D(I, AN) > t + s$, so that $N_1 = N$ and the error is corrected as claimed. Similarly, if $t < W(E) \leq t + s$, then for any other code point AN', $D(I, AN') > t$ so that no code point within distance t of I exists and a detected error will be announced as claimed.

Conversely, suppose that $D_{min} \leq 2t + s$. Then, there exist distinct code points AN and AN' such that $D(AN, AN') \leq 2t + s$.

Let $AN - AN' = E' - E$, where E' is the sum of the t highest-order terms in the NAF of $AN - AN'$ and $-E$ is the sum of the remaining terms. Hence, $W(E') \leq t$ and $W(E) \leq t + s$. Now, if the actual result is

$$I = AN + E = AN' + E'$$

a decoder to perform the claimed correction and detection would be inconsistently required to correct E' and also to correct or detect E. Hence, a decoder to perform the claimed correction and detection does not exist, and the theorem is proved.

Since our interest in this chapter is the correction of errors in computer arithmetic, we shall not further consider AN-codes for correcting errors in unrestricted integer arithmetic.

B. *ARITHMETIC MODULO m*

It has been pointed out in Chapter II that most digital computers perform modulo m arithmetic, where m is either 2^n (two's complement arithmetic) or 2^n-1 (one complement arithmetic). For any positive integer m, the residue of an integer I, modulo m is defined to be the remainder resulting from the division of I by m, and will be denoted hereafter as $R_m(I)$. In other words, $R_m(I)$ is the unique integer in the range $0 \leq R_m(I) < m$ such that $I - R_m(I)$ is a multiple of m. For example, $R_7(23) = R_7(-5) = 2$. For our purposes, the two most important properties of residues are

$$R_m(I_1 \pm I_2) = R_m\,[c_1 m + R_m(I_1) \pm c_2 m \pm R_m(I_2)] =$$
$$R_m\,[(c_1 \pm c_2)m + R_m(I_1) \pm R_m(I_2)] =$$
$$R_m\,[R_m(I_1) \pm R_m(I_2)] \qquad (3.2)$$

Similarly

$$R_m(I_1\ I_2) = R_m\left\{[c_1 m + R_m(I_1)]\,[c_2 m + R_m(I_2)]\right\}$$
$$= R_m\,[R_m(I_1) \cdot\ R_m(I_2)] \qquad (3.3)$$

Those properties say in essence that if only the residue of the final result is of interest, then the residues of all intermediate results can be taken without changing the final result. The set of integers $Z_m = \{1, 2, \ldots m-1\}$ form a ring under modulo m addition, which we denote by $\oplus$, and modulo m multiplication, thereafter denoted by $\otimes$.

So, $I_1 + I_2 = R_m(I_1 \oplus I_2)$ is the ring sum and

$$I_1 \otimes I_2 = R_m(I_1 \otimes I_2) \qquad (3.4)$$

is the ring multiplication. The ring above identified is called the ring of integers modulo m, and the ring operations are arithmetic operations modulo m.

In a digital computer the arithmetic unit performs arithmetic modulo m, and the actual results of such operations are also integers in Z_m. Moreover, if the number of code points is B, it is convenient to choose m = AB. Then, if AN_1 and AN_2 are any code points, and hence necessarily integers in Z_m, we have

$$AN_1 \oplus AN_2 = R_{AB}[AN_1 + AN_2] = AR_B(N_1 + N_2)$$

and since $0 \leq R_B(N_1 + N_2) < B$, it follows that the modulo m sum of any two code points is another code point. Thus, an AN-code is truly linear for modulo m addition. The set of code points is closed under addition modulo m and thus the code is a group; besides, the modulo m product of a code point with any integer in Z_m is another code point so that the code is in fact an ideal in Z_m.

If the code point AN is the correct result of some operation in Z_m, but the actual result is I, we define the ring error F as the unique element of Z_m such that $I = AN \oplus F$, or equivalently, $F = I \ominus AN$. Note that the actual error E is defined as $I = AN + E$ so that the actual error E is either F or $F - m$ depending on whether $E \geq 0$ or $E < 0$, respectively. Note also that the actual error may not be an integer in Z_m.

The syndrome S(I) associated with the possibly erroneous result $I = AN \oplus F$ is defined to be the residue of I modulo A, the generator of the code, i.e., $S(I) = R_A(I)$. Since $m = AB$, we have

$$S(I) = R_A(I) = R_A(AN \oplus F) = R_A(F) \tag{3.5}$$

and we see that the syndrome depends only on the ring error which has occurred, and we commonly speak of $R_A(F)$ as the syndrome of the ring error F. The importance of the syndrome is that it uniquely identifies which ring errors are consistent with the result I.

<u>Lemma 3.1</u>

For any I in Z_m, the set of integers F in Z_m such that $I = AN \oplus F$ for some code point AN is the set of ring errors whose syndrome is S(I). This set is the set of ring errors consistent with I and contains precisely B distinct ring errors.

Proof

The possible ring errors are the distinct F_i such that

$$I = Ai \oplus F_i, \qquad 0 \leq i < B$$

so there are precisely B such ring errors. Moreover $S(I) = R_A(Ai + F_i) = R_A(F_i) = S(F_i)$ for all of these B ring errors. The A different possible values for S(I) together with the B ring errors as given above with the syndrome S(I) account for all $m = AB$ ring errors in Z_m, so no ring errors except the F_i given above can have the same syndrome S(I).

In coding for arithmetic in Z_m, one generally first specifies a set T_c of ring errors which one desires to correct. That is, for an error F in T_c and a result such that $I = AN \oplus F$, one wishes the decoder to correct the result I to its true value AN. In practice, one would always assign $F = 0$ to T_c since one would certainly wish the decoder to pass correct results unchanged to the user, but this assumption is not necessary in the following theory. Similarly, one generally specifies another set T_d of ring errors (which may be the empty set) disjoint from T_c which one desires to detect. That is, for an error F in T_d and a result I such that $I = AN \oplus F$, one wishes the decoder to announce a detected but uncorrectable error. The following theorem gives the necessary and sufficient conditions for determining whether such error correction and detection is possible for a given AN-code.

Theorem 3.3

Let T_c and T_d be any disjoint subsets of Z_m. Then, an AN-code (with $m = AB$) can correct all ring errors in T_c in its code points and can detect all ring errors in T_d in its code points if and only if there is no ring error F_1 in T_c which has the same syndrome as some other ring error F_2 in either T_c or T_d.

Proof

If the distinctness condition on the syndromes holds, and if $I = AN \oplus F$ for some F in T_c, then Lemma 3.1 implies that no other error F' in T_c or T_d is possible so that I can be unambiguously decoded to AN or a detected error unambiguously announced if there is no such F in T_c. Conversely, if the distinctness condition does not hold, i.e., if $S(F_1) = S(F_2)$ for some F_1 in T_c and some other F_2 in T_c or T_d, then by Lemma 3.1 both F_1 and F_2 are possible errors for the result $I = F_1$ and hence a decoder would be inconsistently required to correct F_1 and to correct or detect F_2.

The above proof for Theorem 3.3 indicates a possible procedure for correcting error in modulo m addition with AN coded operands. First, one could compute the syndrome S(I) from the actual result I, and then, if an error F in T_c has occurred, to look up its value in a table of syndromes for the ring errors in T_c. A detected error is announced if S(I) does not appear in the table. After F has been found, the result I is corrected to the value $I \ominus F$. If the actual result of an arithmetic operation is

$$I = AN + E = AN \oplus F$$

the actual error E is either F if $E \geq 0$, or F - m if $E < 0$. In consequence the arithmetic weight of the actual error is either W(F) or W(-F) since $W(F-m) = W(m-F)$. W(E) has been previously shown to be an appropriate measure of the adder malfunction for modulo m addition with $m = 2^n-1$ or $m = 2^n$. We study next a way of measuring ring error which was proposed by Ras and Garcia.

Definition 3.2

The modular weight of an integer F thereafter denoted $W_m(F)$ is the minimum between W(F) and W(-F). It is apparent from the above definition that

$$W_m(F) \leq W(F).$$

It is easily verified that the modular weight, for any m, satisfies two of the three norm properties, namely

$$W_m(F) = W_m(-F) \qquad (3.6)$$

and

$$W_m(F) \geq 0 \text{ equality holding only if } F = 0. \qquad (3.7)$$

The triangle inequality

$$W_m(F_1 \oplus F_2) \leq W_m(F_1) + W_m(F_2) \qquad (3.8)$$

does not hold, however, for arbitrary m. Nevertheless, it does hold for the cases of greatest interest, namely $m = 2^n-1$, $m = 2^n$ and $m = 2^n+1$, as we will show in the following theorem due to Garcia. The proof of this theorem is lengthy and tedious. But in face of its importance a complete proof will be presented for the case of $m = 2^n-1$. This proof gives some insight into why the triangle inequality fails for arbitrary m, and may easily be changed for the cases where $m = 2^n$ or $m = 2^n+1$.

Theorem 3.4

For any integers F_1 and F_2 in Z_m, the triangle inequality (3.8) for modular weight is satisfied if m is any of the moduli 2^n-1, 2^n, or 2^n+1.

Proof

Suppose first that $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(-F_2)$. If also $F_1 + F_2 < m$, we have

$$W_m(F_1 \oplus F_2) = W_m(F_1 + F_2) \leq W(m - F_1 - F_2)$$

and thus by the triangle inequality for arithmetic weight (2.7),

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(m - F_2) = W_m(F_1) + W_m(F_2)$$

so the theorem is proved for this case. Similarly, if $F_1 + F_2 \geq m$,

$$W_m(F_1 \oplus F_2) = W_m(F_1 + F_2 - m) \leq W(F_1 + F_2 - m)$$

so that again by (2.7) we have

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(m - F_2) = W_m(F_1) + W_m(F_2).$$

This completes the proof for $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(\ominus F_2)$, and by symmetry for $W_m(F_1) = W(\ominus F_1)$ and $W_m(F_2) = W(F_2)$. We note that we have not yet made use of our hypothesis on the form of m.

Suppose, momentarily, that the theorem holds when $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(F_2)$. Then the theorem also holds for $W_m(F_1) = W(\ominus F_1)$ and $W_m(F_2) = W(\ominus F_2)$, since then, by property (3.5)

$$W_m(F_1 \oplus F_2) = W_m[(\ominus F_1) \oplus (\ominus F_2)] \leq W_m(\ominus F_1) + W_m(\ominus F_2)$$
$$= W_m(F_1) + W_m(F_2).$$

Hence, the theorem will be proved if we can prove it for the case where $W_m(F_1) = W(F_1)$ and $W_m(F_2) = W(F_2)$, as we now proceed to do.

If also $F_1 + F_2 < m$, we have

$$W_m(F_1 \oplus F_2) = W_m(F_1 - F_2) \leq W(F_1 + F_2)$$

and again we have, according to (2.7)

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(F_2) = W_m(F_1) + W_m(F_2).$$

If $F_1 + F_2 = m$, we have $W_m(F_1 \oplus F_2) = 0$, so the theorem holds trivially.

It remains to consider the case $F_1 - F_2 > m$. We now impose our restriction on m and suppose first that $m = 2^n-1$. By our hypotheses,

$$m = 2^n-1 < F_1 + F_2 < 2m < 2^{n+1}.$$

It follows from Corollary 2.1 that the NAF of $F_1 + F_2$ is given by the $(n - 2)$-tuple $[b_{n+1}, b_n, \ldots, b_1, b_0]$, where either (b_{n+1}, b_n) is (0,1) or (1,0). In the former case, the term

$+2^n$ appears in the NAF of $F_1 + F_2$, and hence

$$W(F_1 + F_2 - 2^n) = W(F_1 - F_2) - 1.$$

Thus,

$$W_m(F_1 \oplus F_2) = W_m(F_1 + F_2 - m) \leq W(F_1 + F_2 - m)$$
$$= W(F_1 + F_2 - 2^n + 1)$$

can be overbounded using the triangle inequality (2.7) as

$$W_m(F_1 \oplus F_2) \leq W(F_1 + F_2 - 2^n) + W(1) = W(F_1 + F_2).$$

Again using (2.7) we conclude

$$W_m(F_1 \oplus F_2) \leq W(F_1) + W(F_2) = W_m(F_1) + W_m(F_2)$$

Finally, we must consider the case when $(b_{n+1}, b_n) = (1,0)$, i.e., when $+2^{n+1}$ is a term in the NAF for $F_1 - F_2$. Repeating the same steps as above, we find

$$W(F_1 + F_2 - 2m) \leq W(F_1 + F_2) \leq W_m(F_1) + W_m(F_2)$$

But also from (3.6) we have

$$W_m(F_1 \oplus F_2) = W_m[(\ominus F_1) \oplus (\ominus F_2)] = W_m[(m - F_1) \oplus (m - F_2)]$$

Noting that $0 < 2m - F_1 - F_2 < m$, it follows that

$$W_m(F_1 \oplus F_2) = W_m(2m - F_1 - F_2) \leq W(2m - F_1 - F_2)$$
$$= W(F_1 - F_2 - 2_m)$$

where we made use of (2.5) in the last step. It follows that again

$$W_m(F_1 \oplus F_2) \leq W_m(F_1) + W_m(F_2)$$

so that the theorem is proved for the modulus $m = 2^n-1$.

It follows from the norm properties (3.6) through (3-8) that for the moduli 2^n-1, 2^n, and 2^n+1, we can use modular weight to define a distance function for Z_m just as we used arithmetic weight to define a distance function for any set of integers.

Definition 3.3

The modular distance between the integers I_1 and I_2 in Z_m, where m is one of the moduli 2^n-1, 2^n, or 2^n+1, is the modular weight of their difference and will be denoted $D_m(I_1, I_2)$, i.e.,

$$D_m(I_1, I_2) = W_m(I_1 \ominus I_2) \tag{3.9}$$

It follows immediately from (3.6) - (3.8) that the modular distance is a true metric for Z_m satisfying the three defining properties for a metric.

$$D_m(I_1, I_2) = D_m(I_2, I_1) \tag{3.10}$$

$$D_m(I_1, I_2) \geq 0 \text{ with equality holding only for } I_1 = I_2 \tag{3.11}$$

$$D_m(I_1, I_2) \leq D_m(I_1, I_3) - D_m(I_3, I_2) \tag{3.12}$$

For the special cases where the modulo m is 2^n or 2^n+1, the minimum modular distance of an AN code is the minimum of the modular distances between pairs of distinct code points.

Theorem 3.5

For an An-code where $m = AB$ is one of the moduli 2^n-1, 2^n, or 2^n+1, the minimum modular distance and the minimum arithmetic distance coincide.

Proof

We know that, for $N_1 - N_2 = N_3 > 0$

$$D_m(AN_1, AN_2) = W_m[A(N_1 - N_2)] = \min [W(AN_3), W(AN_4)]$$

where $N_4 = B - N_3$ which shows that the modular distance between distinct code points is always the arithmetic weight of some nonzero code point. Hence, the minimum modular distance cannot be smaller than W_{min}, the minimum arithmetic weight of the non-zero code points. Conversely, if AN_5 is a code point of weight W_{min}, then

$$D_m(0, AN_5) \leq W(AN_5) = W_{min}$$

so that the minimum modular distance cannot exceed W_{min}. It follows that the minimum modular distance is exactly W_{min} and hence by Theorem 3.1, exactly equal to the minimum arithmetic distance of the code.

As a consequence, we will use the symbol D_{min} for both of these minimum distances.

A fundamental theorem, analogous to Theorem 3.2 for arithmetic distance is introduced next:

Theorem 3.6

For any $t \geq 0$ and $s \geq 0$, an AN-code where $m = AB$ is one of the moduli 2^n-1, 2^n, or 2^n+1 can correct all ring errors of modular weight $t + s$ or less in its code points if and only if its minimum modular distance satisfies

$$D_{min} > 2t + s.$$

<u>Proof</u>

Suppose that $D_{min} > 2t + s$. We shall show that the decoder which decodes the result I into the nearest code point AN_1 if there is a code point AN_1 with $D_m(I, AN_1) < t$, and announces a detected error otherwise, performs the claimed correction and detection. For, suppose $I = AN \oplus F$, where AN is the correct result and $W_m(F) \leq t$. Then, $D_m(I, AN) \leq t$, but for any other code point AN', we have by the triangle inequality (3.12) that

$$D_m(I, AN') \geq D_m(AN, AN') - D_m(I, AN) > t + s$$

so that $N_1 = N$ and the error is corrected as claimed. Similarly, if $t < W_m(F) \leq t + s$, then for any other code point AN',

$$D_m(I, AN') > t$$

so that no code point within distance t of I exists and a detected error will be announced as claimed.

Now, suppose that $D_{min} \leq 2t + s$. Then, there exist distinct code points AN and AN' such that

$$D(AN, AN') \leq 2t - s.$$

Let

$$AN - AN' = E' - E$$

where E' is the sum of the t highest-order terms in the NAF of AN - AN' and - E is the sum of the remaining terms. (Note that E or E' may not be an integer in Z_m.) Define the ring errors F and F' as $F = R_m(E)$ and $F' = R_m(E')$. The same type of argument used in the latter part of the proof of Theorem 3.4, i.e., consideration of the NAF;s of E and E', shows that for any of the moduli 2^n-1, 2^n, or 2^n+1, $W_m(F) \leq W(E) \leq t + s$ and $W_m(F') \leq W(E') \leq t$. But it follows also from the property (3.2) of residues that

$$AN' \ominus AN = R_m(AN' - AN) = R_m(E - E') = F \ominus F'$$

and hence that

$$AN \oplus F = AN' \oplus F'$$

where F and F' are distinct ring errors with $W_m(F) \leq t + s$ and $W_m(F') \leq t$. Thus, if the actual result of some operation is

$$I = AN \ominus F = AN' \oplus F'$$

then a decoder to perform the claimed correction and detection would be inconsistently required to correct F' and also to correct or detect F, and hence does not exist, which proves the theorem.

C. PERFECT AN SINGLE ERROR CORRECTING CODES

The sphere of radius t about a point I in Z_m is defined as the set of all integers J in Z_m such that $D_m(I,J) \leq t$. The volume of such sphere is the number of integers in the sphere. The volume of a sphere is also equal to the number of integers F in Z_m with $W_m(F) \leq t$, since $J = I \oplus F$ is in the sphere if and only if $W_m(F) \leq t$. The volume of a sphere is independent of its center I.

Suppose an AN-code with $m = AB$ can correct all ring errors F of modular weight t or less. Then the spheres of radius t about each of its code points must be disjoint since otherwise there would be a possible result I at modular distance t or less from two code points so that one of the two corresponding errors of modular weight t or less could not be corrected. But since there must be B such disjoint spheres each with V_t points and also there are only a total of m integers in Z_m, it follows that any AN-code capable of correcting t or fewer errors must satisfy the inequality

$$BV_t \leq m = AB \text{ or}$$

$$V_t \leq A \qquad (3.13)$$

which is a bound corresponding precisely to the well known Hamming bound for parity check error correcting codes. An AN-code is said to be a perfect code or a sphere packed code if the bound in equation (3.13) holds with equality. Consider an AN-code with generator is a prime number $A = p$. Then the set of integers modulo A is a field G(p) under addition and multiplication modulo A. Before proceeding to the presentation of the next theorem, let's recall that in a field G(p) if a is a primitive, then

$$a^0 = 1$$
$$a^1 = a$$
$$a^2 = a \otimes a$$
$$a^n = a^{n-1} \otimes a \qquad n < p$$
$$a^{p-1}$$

are all distinct elements of G(p). Also to simplify the notation in the proof to be given next, recall that in arithmetic modulo p, an integer is represented by $R_p(I)$. So $p = R_p(p) = 0$. Also, the additive inverse of $a < p$ is

$$-a = R_p(-a) = p-1.$$

Theorem 3.7

If A is an odd prime p and if 2 is a primitive in G(p) then:

(a) A generates a code with $D_{min} = 3$ and $B = 2^n+1$ where $n = (1/2)(A-1)$.

(b) Also, if $R_p(-2) = p-2$ is a primitive in G(p) but 2 is not, then A generates a code with same distance and $B = 2^n-1$.

(c) The codes given by this theorem are perfect codes.

Proof

If A is prime, then

$$2^{A-1} - 1 = (2^{(A-1)/2} + 1) \times (2^{(A-1)/2} - 1) = 0.$$

If 2 is a primitive then $2^{(A-1)/2} \neq 1$ and so

$$2^{(A-1)/2} + 1 = 0 \text{ or } 2^{(A-1)/2} = -1.$$

As a consequence, the consecutive powers of 2

$$1,\ 2,\ 2^2,\ \ldots,\ 2^{(A-3)/2},\ 2^{(A-1)/2} = -1,\ -2,\ \ldots,\ -2^{(A-3)/2}$$

are all distincts. In fact, they are all nonzero elements in G(p). So all possible weight F error have distinct syndromes, and by Theorem 3.3 part (a) of the theorem holds. If $\ominus 2$ is primitive, then $1, 2, 2^2 \ldots 2^{(A-3)/2}$ must be distinct since their squares are. Also none is equal to $\ominus 1$, for if $2^j = \ominus 1$ for some $j \leq (A-3)/2$, then $2^{2j} = 1 = (\ominus 2)^{2j}$ for some $2j \leq A-3$, which is impossible if $\ominus 2$ is primitive. Again,

$(2^{(A-1)/2} - 1)\ (2^{(A-1)/2} + 1) = 0$. If ((A-1)/2 is even,

$2^{(A-1)/2} = (\ominus 2)^{(A-1)/2} \neq 1$; hence $2^{(A-1)/2} = -1$. Then, 2 is primitive and part a applies. If, on the other hand, $\ominus 2$ is primitive and 2 is not, then (A-1)/2 must be odd. Then, $2^{(A-1)/2} - 1 = 0$, and the syndrome of the weight-1 errors 1, $\ominus 2,\ 2^2,\ \ldots,\ 2^{(A-3)/2},\ \ominus 1,\ +2,\ \ominus 2^2,\ \ldots,\ \ominus 2^{(A-3)/2}$ are distinct. By Theorem 3.3 part b holds.

Conversely, Theorem 3.3 implies that if a or b holds, all the residues mod A must be congruent to $\pm 2^i$, and A must be odd. Then 2, and therefore all the residues mod A, are relatively prime to A, so that A must be prime. Then the residues mod A form a field. The order of 2 is at least (A-1)/2 since the smaller powers of 2 have distinct residues. But the order of 2 must divide A-1, so it must be either A-1 or (A-1)/2. The same is true of $\ominus 2$. If the orders of both 2 and $\ominus 2$ are (A-1)/2, then $\pm 2^j$ is always an even power of a. This is impossible. Therefore, either 2 or $\ominus 2$ must have order A-1, that is, be primitive. It remains to be shown that the code is perfect.

The errors of weight 1 or less are just the integers 0 or 2^i and

$$2^n - 2^j - 1 \text{ for } 0 \leq j < n.$$

So there are 2n + 1 integers, and the volume of radius 1 sphere in Zm is

$$V_1 = 2n + 1 = 2(\frac{A-1}{2}) + 1 = A.$$

Table I presents some codes given by Theorem 3.7.

Table I

Some Perfect, Single Error Correcting AN Codes

A	B	M	BITS (N)	BITS (AN)	CLASS
11	3	33	2	5	N
13	5	65	3	6	N
19	27	513	5	9	N
23	89	2047	7	11	C
29	565	16385	10	14	N
37	7085	262145	13	18	N
47	178481	8388607	18	23	C
53	1266205	67108865	21	26	N
59	9099507	536870913	24	29	N
61	17602325	1073741825	25	30	N
67	128207979	8589934593	27	33	N
71	483939977	34359738367	29	35	C
79	6958934353	549755813887	33	39	C

```
R; T=0.52/0.94 16.08.07
```

A = code generator

B = number of code points

M = A x B = modulo of arithmetic performed by adder

BITS (N) = number of bits required to represent the largest integer (B-1)

BITS (AN) = number of bits required to represent the largest code point

IV. CYCLIC AN CODES

In this chapter we will study a very important class of arithmetic codes: the cyclic AN codes. The cyclic nature of some AN codes was apparently first noticed by Mandelbaum in 1967.

The radix-two form of a code point is of great interest, since it is the way code points are represented in digital computers. By analogy with parity check error correcting codes we will thereafter refer to this representation as a code word. Since a code word is a representation of a code point we will use the two expressions as equivalents whenever it seems appropriate.

Definition 4.1

The cyclic shift $T(N)$ of a number $N \varepsilon Z_m$ with radix-two form is $[a_{n-1}, a_{n-2}, \ldots, a_1, a_0]$ is another number $N_s \varepsilon Z_m$ which is represented in radix-two form by the n-typle $[a_{n-2}, a_{n-3}, \ldots, a, a_0, a_{n-1}]$.

Definition 4.2

A cyclic AN code is an AN code which is closed under cyclic shifting. Consider a cyclic AN code with a code point $AN_1 \varepsilon Z_m$ such that its corresponding code word is of the form 111...111. This code is said to include an all one code word, and has $D_{min} \leq 2$ since the distance between the all one code word and the 0 word is two. Such a code is therefore of little interest. But the all one code word necessarily corresponds to the largest code point. As a consequence, if we remove this code point from the code the remaining set of numbers is still a cyclic AN code. In this chapter we will only consider those cyclic AN codes which do not include an all one code word.

In Chapter II we have discussed the carry propagation in addition modulo m. We have verified then, that in addition modulo $m = 2^n-1$, the carry propagates from the most significant binary position into the least significant binary position. Therefore, for modulo $m = 2^n-1$

$$T(AN) = AN \oplus AN = 2 \otimes AN \tag{4.1}$$

what implies that a AN code such that $AB = 2^n-1$ is closed under cyclic shifting and therefore cyclic. This observation is expressed in the following theorem.

Theorem 4.1

An AN code with B code points is cyclic if and only if A generates an ideal in Z_m, the ring of integers modulo $M = AB = 2^n-1$.

Proof

The sufficiency of the condition given by the theorem has been proved before by showing that

$$T(AN) = 2 \otimes AN \varepsilon Z_m$$

Conversely, if a cyclic AN code includes the code word

$x = s_{n-1}, x_{n-2}, \ldots, x, x_0 = AK$ then

$y = x_{n-2}, x_{n-3}, \ldots, x_1, x_0, x_{n-1}$ is also a code word

and hence a multiple of A.

$$y = \sum_{i=0}^{n-2} x_i 2^{i+1} + a_{n-1} = \begin{cases} 2AK, & x_{n-1} = 0 \\ 2AK-(2^n-1), & x_{n-1} = 1 \end{cases}$$

For y to be a multiple of A it is required that A divides 2^n-1. Hence A generates an ideal in Z_m.

As a consequence of Theorem 4.1 we will assume that the modulus m is given by m = AB = 2^n-1, throughout the remainder of our discussion on cyclic AN codes. To represent integers modulo 2^n-1, n binary digits are required. It is therefore natural to think of the integer n as the code length. Again borrowing the terms from the theory of error correcting codes used in communications, we define the code rate R to be the ratio of the base two logarithm of the number of code points to the base two logarithm of the total number of integers in the ring Z_m, i.e.,

$$R = (\log_2 B)/(\log_2 m) = (\log_2 B)/(\log_2 A + \log_2 B) \quad (4.2)$$

The quantity $\log_2 A$ is called the redundancy of the code, and is an approximate measure of the number of extra binary portions required to represent the code points as opposed to those required to represent the original integers without coding.

Suppose now that a cyclic AN code has $D_{min} = W_{min} = 1$. Then, 2^i must be a code word for some i, $0 \leq i \leq n$, and hence 1 must be a code point since 2^i is i cyclic shifts of 1. It follows that A must be 1 and we have proved:

<u>Theorem 4.2</u>

Every cyclic AN code with A > 1 had $D_{min} \geq 2$. The codes with A = 3 are the least-redundant codes satisfying Theorem 4.2 and have D_{min} exactly 2 for B > 1 since W(A) = 2. Noting that

$3 = 2^2-1$ divides 2^n-1 if and only if n is even, we have as an immediate consequence:

Corollary 4.1

For every even n greater than two, $A = 3$ generates a cyclic AN-code with $B = (2^n-1)/3$ code points and $D_{min} = 2$. These codes can detect all single modular errors in their code points.

We show next that there is a natural length n, determined entirely by A, for cyclic AN codes generated by A. We first remark that Theorem 4.1 shows that A must be odd if A generates a cyclic AN code.

For any positive odd integer A, the exponent of 2 modulo A, denoted e(A), is the least positive integer i such that A divides 2^i-1. From (3.2), we see that an equivalent statement is that i is the least positive integer such that $R_A(2^i) = 1$. We show first that e(A) is always well-defined. Consider the sequence of numbers $R_A(2^j)$, $j = 1, 2, 3, \ldots$. Since A is odd, it cannot divide 2^j for any j, so that these numbers are all between 1 and A-1 inclusive. But then there must be specific integers j and k with $1<j<k$ A such that $R_A(2^j) = R_A(2^k)$. By (3.2), this is equivalent to

$$R_A[2^j(2^{k-j} -1)] = 0$$

which implies that A divides $2^{k-j} -1$, since A is odd. This proves that e(A) is always well-defined for every positive odd integer and also that $e(A) \leq A -1$. A continuation of this argument shows further that the sequence of residues $R_A(2^j)$, $j = 1, 2, 3, \ldots$, is periodic with period e(A) and hence that A divides 2^n-1 if and only if n is a multiple of e(A).

If A generates a cyclic AN code of length n, it follows from Theorem 4.1 that A divides 2^n-1 and hence that n is a multiple of e(A). If $n > e(A)$, then $2^{e(A)}-1$ is a code point since it is a multiple of A less than $AB = 2^n-1$. Hence (assuming $A > 1$), the AN code has $D_{min} = 2$ since it has a code point of arithmetic weight two. In view of the fact that, except for $A = 3$, cyclic AN codes with $D = 2$ are of little interest, we will adopt the convention that, unless otherwise specified, the length n of the cyclic AN code generated by an odd positive integer A is to be taken as e(A), the exponent of 2 modulo A.

Theorem 4.3

The cyclic AN code generated by an odd integer A $(A > 1)$ has $D_{min} \geq 3$ if and only if either

(a) e(A) is odd, or

(b) e(A) is even but A does not divide $2^{e(A)/2}+1$.

Proof

The only possible code points with arithmetic weight two are 2^i-1 and 2^i+1 for $0 < i < n = e(A)$. Since $i < e(A)$, 2^i-1 is not divisible by A and hence is not a code word. If 2^i+1 is a code point and hence divisible by A, then $2^{2i}-1 = (2^i+1)(2^8-1)$ is also divisible by A, so that 2i must be a multiple of e(A). Since $i < e(A)$, this is impossible if e(A) is odd and requires $e(A) = 2i$ if e(A) is even. Thus, $2^{e(A)/2}+1$ is the only possible code point with arithmetic weight two and will be an actual code point if and only if it is divisible by A. This proves the theorem.

Example

Take A = 21. We have then $R_A(2^j) = 2, 4, 8, 16, 11, 1$ for $j = 1, 2, 3, 4, 5, 6$. Hence $n = e(A) = 6$ for the cyclic AN code generated by A = 21. We note that e(A) is even but A = 21 does not divide $2^{e(A)/2} + 1 = 9$, so that $D_{min} \geq 3$ by Theorem 4.3. In fact, $W(21) = W(2^4 + 2^2 + 2^0) = 3$, so that $D_{min} = 3$. This single-error correcting code has $B = (2^n-1)/4 = 3$ code points.

We are now in a position to recognize that the codes given by Theorem 3.7 (b) are cyclic codes. In fact, they are the best, less redundant, cyclic AN codes with distance 3. The codes listed in Table I and identified with C are of this category.

A. NEGACYCLIC AN CODES

It is interesting at this point to open a brief parenthesis to a comment on another class of codes related to that of cyclic AN codes, the so called negacyclic AN codes. We have shown that in addition modulo $m = 2^n-1$ the carry generated is the highest binary position propagates to the lower position. We have also related this observation with the cyclic character of AN codes with $m = AB = 2^n-1$. Similarly in addition modulo $m = 2^n+1$ we can easily see that the carry generated at the highest binary position is subtracted from the lowest binary position. This fact gives the negacyclic character of AN codes such that $m = AB = 2^2 + 1$. The codes given by Theorem 3.7(a) are of this category. Also some of those codes with $D_{min} = 3$ are listed in Table I, where marked with N.

B. CALCULATION OF D_{min} FOR CYCLIC AN CODES

In this section we will verify that the search for the code point with minimum arithmetic weight in a cyclic AN code can be limited to one sixth of its code points. First, notice from Corollary 2.1 that every nonzero $I \in Z_m$ for $m = 2^n-1$ has an (n+1) position NAF $[b_n, b_{n-1}, \ldots, b_1, b_0]$ and also that from Theorem 2.2.

$$(b_n, b_{n-1}) = (0,0) \quad \text{for } 0 < 3I < 2^n \tag{4.3}$$

$$(b_n, b_{n-1}) = (0,1) \quad \text{for } 2^n < 3I < 2^{n+1} \tag{4.4}$$

$$(b_n, b_{n-1}) = (1,0) \quad \text{for } 2^{n+1} < 3I \tag{4.5}$$

This suggests separting the set of integers in Z_m into the corresponding three disjoint subsets characterized by the two highest coefficients in their NAF's. For simplicity, we shall now adopt an interval notation for sets of integers and write (a,b) or (a,b] to denote the set of integers I such that $a < I < b$ or $a < I \leq b$, respectively, where a and b are any real numbers. In particular, the sets of integers

$$L3m = (0,\ 2^n/3) \tag{4.6}$$

$$M3m = (2^n/3,\ 2^{n+1}/3) \tag{4.7}$$

$$U3m = (2^{n+1}/3,\ 2^n) \tag{4.8}$$

are precisely the sets of integers $I \varepsilon Z_m$ satisfying (4.3), (4.4) and (4.5) respectively, and hence our desired partition of the nonzero integers in Z_m.

<u>Lemma 4.1</u>

For an integer $I \varepsilon Z_m$

$$W_m(I) = \begin{cases} W(I) & \text{if } I\varepsilon M3m \text{ and } I \text{ is even, or if } I\varepsilon L3m. \\ W(m-I) & \text{otherwise.} \end{cases}$$

<u>Proof</u>

<u>Case A, $I\varepsilon L3m$.</u>

By (4.3) the NAF of I has two leading zeros so that $W(2^n-1) = W(I) +1$. Thus, $W(m-I) = W(2^n-I-1)$ $W(2^n-I) - w(1) =$ $W(I)$ where the inequality follows from the triangular inequality for arithmetic distance (2.7) since distance is just the weight of the difference. Hence $W_m(I) = W(I)$ as claimed.

<u>Case B, $I\varepsilon U3m$.</u>

Now $(m-I) = 2^n -1 -I$ belongs to L3m so that $W_m(I) = W(m-I)$ by Case A above.

<u>Case C, $I\varepsilon M3m$.</u>

Now $W(I) = W(2^n-I)$. If I is odd, then 2^n-1 is also odd and in M3m. Since 2^n-I is odd, subtracting one cannot increase its weight so we have $W(2^n-I-1) \leq W(2^n-I) = W(I)$ and hence $W_m(I) =$ $W(m-I)$. If I is even, then 2^n-1-I is odd and in M3m. The same argument then gives $W_m(2^n-1-I) = W(K)$ but $W_m(m-I) = W_m(I)$ so $W_m(I) = W(I)$. Thus, the theorem is proved.

The subsequent discussion will be easier if we use the following Lemma, due to Hartman.

Lemma 4.2

For an integer $I \varepsilon Z_m$

$$W_m(I) = W_m[T(I)].$$

Proof

Case A, $I<2^{n-1}$.

If $I\varepsilon L3m$, then either $T(I)\varepsilon L3m$ or $T(I)\varepsilon M3m$ is even. In either case $W_m[T(I)] = W[T(I)] = W(I) = W_m(I)$. If $I\varepsilon M3m$ and is even then $W_m[T(I)] = W(2^n-1-2I) = W(2^n-2I) + 1$ since the NAF of $2^n - 2I$ ends in two zeros. Since $2I\varepsilon U3m$, $W(2^n -2I) = W(2I) -1$. *Hence by Lemma* 4.1 $W_m[T(I)] = W(2^n-1-2I) = W(I) = W_m(I)$.

Suppose finally that $I\varepsilon M3m$ and I is odd. Then

$$W(2^n-I) = W(I) \text{ and } W(2^n-2I) = W(I) - 1.$$

There are only two possible types of endings for the NAF of I. They are either 0, -1, followed by any number (including none) of repetitions of 0, 1; or 0, 0, 1, followed again by any number (including none) of repetitions of 0, 1. With the first type of ending we have:

$$W(2^n-I-1) = W(2^n-I) = W(I) - 1 \text{ and}$$
$$W(2^n-2I-1) = W(2^n-2I) = W(I) - 1.$$

With the second type of ending we have:

$$W(2^n-I-1) = W(2^n-I) = W(I) \text{ and}$$
$$W(2^n-2I-1) = W(2^n=2I) +1 = W(I).$$

Thus for either type of ending we have:

$$W_m(I) = W(2^n-I-1) = W(2^n-2I-1) = W_m[T(I)].$$

Case B, $I \geq 2^{n-1}$.

Let $J = 2^n-1-I$.

Now $W_m[T(I)] = W_m[m-T(I)]$

$= W_m[2^n-1-(2I-2^n+1)]$

$= W_m[2J]$.

But $W_m(2J) = W_m(J)$ since $J < 2^n-1$ and by Case A above.

Hence $W_m[T(I)] = W_m(J) = W_m(I)$ which completes the proof.

Lemmas 4.1 and 4.2 give us the tools to derive the following theorem.

Theorem 4.5

In every cyclic AN code there is always at least one code point AN_1 in L3m with N_1 odd such that $W(AN_1) = W_{min}$.

Proof

For any AN_2 in U3m, - AN_2 = m - AN_2 is in L3m and has the same modular weight. If AN_2 is in M3m and is odd, then - AN_2 is also in M3m and is even and has the same modular weight. Finally, for AN_2 in M3m and even, AN_2 is the cyclic shift (and hence by Lemma 4.2 has the same modular weight) as $AN_2/2$, which is in L3m. Similarly, the even code points in L3m are some cyclic shift, and hence, have the same modular weight, of odd code points in L3m. Thus, there is always a nonzero odd code point in L3m with minimum modular weight. But by Lemma 4.1 modular and arithmetic weights coincide in L3m. Hence, there is always a nonzero odd code point in L3m whose arithmetic weight is also the minimum modular weight in the code. But this minimum modular weight coincides with the minimum arithmetic weight W_{min} since the modular weight of any code point is at most its arithmetic weight.

Theorem 4.5 shows that the search for the minimum weight code point in a cyclic AN code can be restricted to one sixth of the code points, namely those odd code points in L3m. Let us now take a deeper look into the properties of the code points in L3m. We begin by noting that the NAF's of code points in L3m, according to (4.3), have n-place NAF's $[b_{n-1}, \ldots, b_1, b_0]$ which are cyclically nonadjacent, that is $b_{n-1}\, b_0 = 0$. Thus if these NAF's are cyclic-shifted, then the result is always again an n-place NAF--although it may be the NAF of an integer not in L3m nor even in Z_m since after some cyclic shifts, the NAF might have -1 as its highest nonzero coefficient.

Suppose now that $I \varepsilon L3m$. We define J_i, $0 \leq i < n$ as an integer whose n-place NAF is the ith cyclic shift of the NAF for $I = J_0$. J_i is not restricted to be positive. We note that J_{i+1} is obtained from J_i by first doubling J_i and then, if the leading coefficient in the NAF for J_i were nonzero, subtracting or adding $m = 2^n-1$ according as this coefficient were +1 or -1. It follows that $R_m(J_{i+1}) = R_m(2J_i)$ and, with the aid of (3.3) that

$$R_m(J_i) = R_m(2^i J_0) = R_m(2^i I),\ 0 \leq i < n \qquad (4.9)$$

Noting next that J_i has an n-place NAF and hence a magnitude less than $m = 2^n-1$, it follows from (4.9) that

$$R_m(2^i I) = \begin{cases} J_i & \text{for } J_i > 0 \\ J_i + m & \text{for } J_i < 0 \end{cases} \tag{4.10}$$

Next, we observe that if $[b_{n-1}, \ldots, b_1, b_0]$ is the NAF of $I = J_0$, then b_{n-1-i} is the leftmost digit in the n-place NAF for J_i. But according to (4.3) this leftmost digit will be +1 if and only if $J_i = R_m(2^i I)$ is in M3m and this leftmost digit will be +1 if and only if $-J_i = m - R_m(2^i I)$ is in M3m. Now from (4.6) we see that for any K in Z_m, K and m - K are either both in M3m or neither is in M3m. It follows that are either both in M3m or neither is in M3m. It follows that b_{n-1-i} is nonzero if and only if $R_m(2^i I)$ lies in M3m. But the arithmetic weight of I, by Theorem 2.1 is just the number of nonzero digits in the NAF of I, so that we have proved:

Theorem 4.6

For every integer I in L3m, its arithmetic weight W(I) is equal to the number of residues $R_m(2^i I)$, $0 \leq i < n$ which lie in $M3m = (2^n/3, 2^{n+1}/3)$.

Application of Theorem 4.6 to the calculation of the arithmetic weight of a code point AN_1 in L3m is facilitated by noting that since m = AB, then

$$R_m(2^i AN_1) = R_{AB}(2^i AN_1) = AR_B(2^i N_1) \tag{4.11}$$

so that $R_m(2^i AN_1)$ lies in M3m if and only if $AR_B(2^i N_1)$ lies in M3m, which in turn occurs if and only if $R_B(2^i N_1)$ lies in the set of integers (B/3, 2B/3]. We have then as an immediate corollary of Theorem 4.6:

Corollary 4.2

For any cyclic AN-code and any code point AN_1 in L3m, $W(AN_1)$ is equal to the number of residues $R_B(2^i N_1)$ which lie in the interval (B/3, 2B/3].

C. MANDELBAUM-BARROWS CODES

The class of equidistant cyclic AN codes to be discussed in this section is of great importance in the development of the theory of AN codes. Besides the fact that they were the first systematically constructed class of AN codes with $D_{min} > 3$, they were the first AN codes to be recognized as cyclic.

In addition they constitute an important step in the attempt to establish analogies between the theory of AN codes and that ofthe parity check error correcting codes. Those codes were discovered independently by Mandelbaum and Barrows. Mandelbaum was the only one who noticed their cyclic nature, while Barrows calculated their exact minimum distance.

Let's define INT(r), where r is a positive real number as the integer part of r. The formulation of Mandelbaum-Barrows code is next made in the following theorem.

Theorem 4.7

If B is any prime such that 2 is primitive in GF(B), then $A = (2^{B-1} - 1)/B$ generates an equidistant cyclic AN-code of length $n = B - 1$ and minimum distance $D_{min} = \text{int}[(B+1)/3]$.

Proof

If B is a prime ρ for which 2 is primitive in GF(p), then

$$e(B) = e(p) = B - 1$$

Let's choose the generator A and the code length such that

$$A = (2^{e(B)} - 1)/B = (2^{B-1} - 1)/B \text{ and}$$

$$n = B - 1.$$

Since 2 is primitive in GF(p) = GF(B), the residues

$$R_B(2^i), \quad 0 \leq i < n = B - 1$$

are all the B - 1 nonzero elements of GF(B). It follows also from (3.4) that for any N_1, $0<N_1<B$, the residues $R_B(2^i N_1)$, $0 \leq i<n = B - 1$, are also all the B - 1 nonzero elements of GF(B) since they are just the modulo B product of the residues in the former set by the fixed nonzero element N_1 in GF(B). It then follows from Corollary 4.2 that all the code words in L3m have arithmetic weight int[(B + 1)/3], and further by Theorem 4.5 that

$$D_{min} = W_{min} = \text{int}[(B + 1)/3]$$

In addition, we know from the proof of Theorem 4.5 that the modular weight of any nonzero code point coincides with the arithmetic weight of a code point in L3m. Hence, all the code points in the entire code, except 0, have modular weight int[(B + 1)/3]. But also the modular distance between any pair of distinct code points is the modular weight of some nonzero code point. Thus, the modular distance between any pair of distinct code points is exactly int[(B + 1)/3], so the code is equidistant.

Table II gives a complete list of Mandelbaum-Barrow codes with n less than 37.

Table II
Mandelbaum-Barrows codes ($n \leq 36$)

B	n	A	D_{min}	Redundancy
3	2	1	1	0
5	4	3	2	1.6
11	10	93	4	6.5
13	12	315	4	8.3
19	18	13,797	6	13.8
29	28	9,256,392	10	23.2
37	36	1,857,283,155	12	30.8

Despite their great theoretical importance Mandelbaum-Barrow codes are highly redundant, since n binary digits are required to represent B = n-1 code points, and therefore, are not attractive for practical applications.

D. INTERMEDIATE DISTANCE CYCLIC AN CODES

We have discussed previously single error, perfect cyclic AN codes, given by Theorem 3.7, and the large distance equidistant Mandelbaum-Barrow codes. It would be of great theoretical importance and also of considerable practical interest if classes of cyclic AN codes with both intermediate distance and redundancy could be found. Unfortunately, there are not at present any systematically constructed classes of cyclic AN codes falling in this category. Nevertheless, some specific codes have been discovered by trial and error techniques. Some of those codes are given in Table III.

TABLE III

Some Intermediate Distance Codes

B	n	A	D_{min}	Redundancy
41	20	25,575	6	14.7
113	28	2,375,535	6	21.8
151	15	217	4	7.8
233	29	2,304,167	8	21.2
241	24	107,415	4	16.1
331	30	3,243,933	6	21.6
337	21	6,223	5	12.6
601	25	55,831	7	15.8
683	22	6,141	4	12.6
1103	29	486,737	7	18.9
1801	25	18,631	5	14.2
2089	29	256,999	6	18.0

E. ANALOGY BETWEEN AN CYCLIC CODES AND CYCLIC PARITY CHECK CODES

To point out the analogies between cyclic AN codes and cyclic parity codes, let's briefly recall the key ideas in the theory of algebraic codes.

With a polynomial $f(X) = f_0 + f_1X + \ldots + f_{n-1} X^{n-1}$ of degree less than n with coefficients in a field F, one associates the vector $f = [f_{n-1}, \ldots, f_1, f_0]$ in the vector space F^n. A parity-check code is simply a set of such vectors which form a subspace of F^n, i.e., which is closed under addition of vectors and scalar multiplication of vectors by the elements in F. The minimum Hamming distance d_{min} of such a code is defined as the least number of positions in which two distinct vectors, or code words, differ. The Hamming weight of a vector—or code word—is the number of its non-zero components. It is verified that the minimum distance d_{min} in the code is always equal to the Hamming weight of the non-zero vector of minimum Hamming weight in the code. The code is cyclic if the cyclic shift of every code word is again a code word. The cyclic shift of f results in a vector whose corresponding polynomial is obtained by multiplying f(X) by X modulo $X^n - 1$. Let g(X) be a monic (i.e., highest coefficient unity) polynomial of minimum degree among the non-zero f(X) in the code. Then, the necessary and sufficient condition for the code to be cyclic is that g(X) divide $X^n - 1$ and that every polynomial multiple of g(X) with degree less than n be a code word and that all code words are of this type. It follows that g(X) is unique in a cyclic code and is called the generator polynomial of the code. The code is therefore an ideal in the ring of polynomials modulo $X^n - 1$.

We recognize, therefore, that if we make 2 correspond to X, g(X) to A, Hamming weight to arithmetic weight, and so on the analogy between cyclic AN codes and cyclic parity check error correcting codes is evident and very strong.

F. THE BCH CONJECTURE FOR AN CODES

The parallels between cyclic AN codes and cyclic parity check error correcting codes, discussed briefly in the previous section, led many theorists to believe that there must exist for cyclic AN codes a bound analogous to the Bose-Chaudhuri-Hocquenghen (BCH) bound for cyclic parity check codes.

We next address very briefly this question.

BCH Bound

If α is a primitive nth root of unity in the extension field E of a field F and g(X) is a polynomial over F dividing x^n-1 and having d-1 consecutive powers of α among its roots, then the minimum Hamming distance of the cyclic parity-check code of length n generated by g(X) is at least d.

To pursue the possibility of a BCH type of bound for AN-codes we find some concepts from algebra and number theory helpful. The cyclotomic polynomial, $Q_i(X)$, is the polynomial having real coefficients whose roots are all and only the primitive ith roots of unity among the complex numbers. The first four $Q_i(X)$ are

$Q_i(X) = x-1$

$Q_2(X) = x+1$

$Q_3(X) = x_2+x+1$

$Q_4(X) = x^2+1$

In general $Q_i(X)$ generated by dividing X^i-1 by the product of all $Q_j(X)$ such that $j<i$ and j divides i, to eliminate the non-primitive roots of -1 from the set of roots of the resulting polynomium.

The integer $A_i = Q_i(2)$ is called the ith cyclotomic number. We note that 2^n-1 has at least as many factors, A_i, as there are factors of n since 2^i-1 divides 2^n-1 if and only if i divides n. Except for $A_2 = A_6 = 3$ $A_i \neq A_j$ when $i \neq j$.

To establish a possible basis for a BCH bound for cyclic AN codes we can factorize the generator A of a length n code as a product of integers A_i, where i divides n, then count the number of consecutive powers of $\alpha = 2^{2\pi i/n}$ among the roots of the corresponding $Q_i(X)$.

The factorization referred above is not unique in general. We must, therefore, find a rule to decide which are the "true" factors of A.

A possible way to resolve this question is as follows:

When there is more than one way to write A as the product of distinct A_i's, i dividing n, and a constant not divisible by any other A_i's where i divides n, then choose the way that yields the largest number of consecutive powers of α among the roots of the corresponding cyclotomic polynomials $Q_i(X)$ associated with the factors A_i. In this setting we always consider $A_1 = 1$ to be one of the distinct factors of A_i of A. In counting consecutive powers of α, α^0, which is a root of $Q_1(X)$, is considered to be adjacent to both α^{n-1} and α^1 since $\alpha^n = \alpha^0$. Let's now state the following.

BCH Conjecture

A cyclic AN-code that has ℓ consecutive powers of α among the roots of the cyclotomic polynomials, $Q_i(X)$ where i divides n, that correspond to the factors A_i in some factorization of A has a minimum arithmetic distance of at least $(\ell + 1)/2$. The BCH conjecture above was proposed by Hartman in 1975, after having proved false a previous conjecture introduced by Chien-Hong in 1970. Hartman's conjecture has been proved valid for some special cases, and also was shown to hold for all cyclic AN codes with length $n \leq 70$, by means of extensive computer search.

V. SEPARATE CODES

A. BASIC CONCEPTS

As pointed out in Chapter I, it is often desirable to have the coded words in an encoding system as consisting of the original set of symbols N, concantenated with a separated set of check symbols C(N)

$$N_c = N \circ C(N)$$

N will be referred to as the information integer, and C(N) as the check integer. If a system like that described in the previous paragraph is used to check the proper operation of an adder, it is also desirable to make sure that the hardware involved in the check processing is independent of the basic adder itself. This will prevent the possibility that a single defective part might affect simultaneously the checker and the adder, resulting in an improperly corrected or even undetected error.

As a consequence of the previous thoughts, the problem of defining an encoding system suitable to the generation of a separate arithmetic code for addition can be stated as follows:

(a) Define a rule to generate C(N) given N, and

(b) Find an operation * such that

$$C(N_1 + N_2) = C(N_1)*C(N_2) \qquad (5.1)$$

Fig. 5.1 represents a block diagram of an adder designed to use a separate code for error correction and/or detection.

The reader is familiar with an error detecting code that meets the above characterization. The well know "nines out proof" for addition, that can be described as follows:

1. Represent the operands by its decimal representation followed by its residue modulo 9 so $C(N) = R_9(N)$.

2. Perform normal addition on the decimal representation of the operands (their information integer). Compute the residue modulo 9 of the result (the check integer of the result).

3. Perform modulo 9 addition (here the operation *) on the check integers of the operands. Compare with the result of 2 for the check integer of the result.

This simple illustration, taken from everybody's elementary school experience, besides being a simple example of separate codes, has the additional virtue of inducing in the reader's mind the idea that, unlike AN codes, separate codes can be found that will be capable of checking both addition and multiplication. This would be a very significant advantage over AN codes, since many arithmetic units are desired to perform multiplication as well as addition. A very good insight into the nature of separate codes is given by Theorem 4.1, due to Peterson.

Theorem 5.1

If there are fewer check symbols than integers in the permissible range of integers, and if the check symbols C(N) satisfy Equation 5.1, then C(N) must be the residue of N mod b

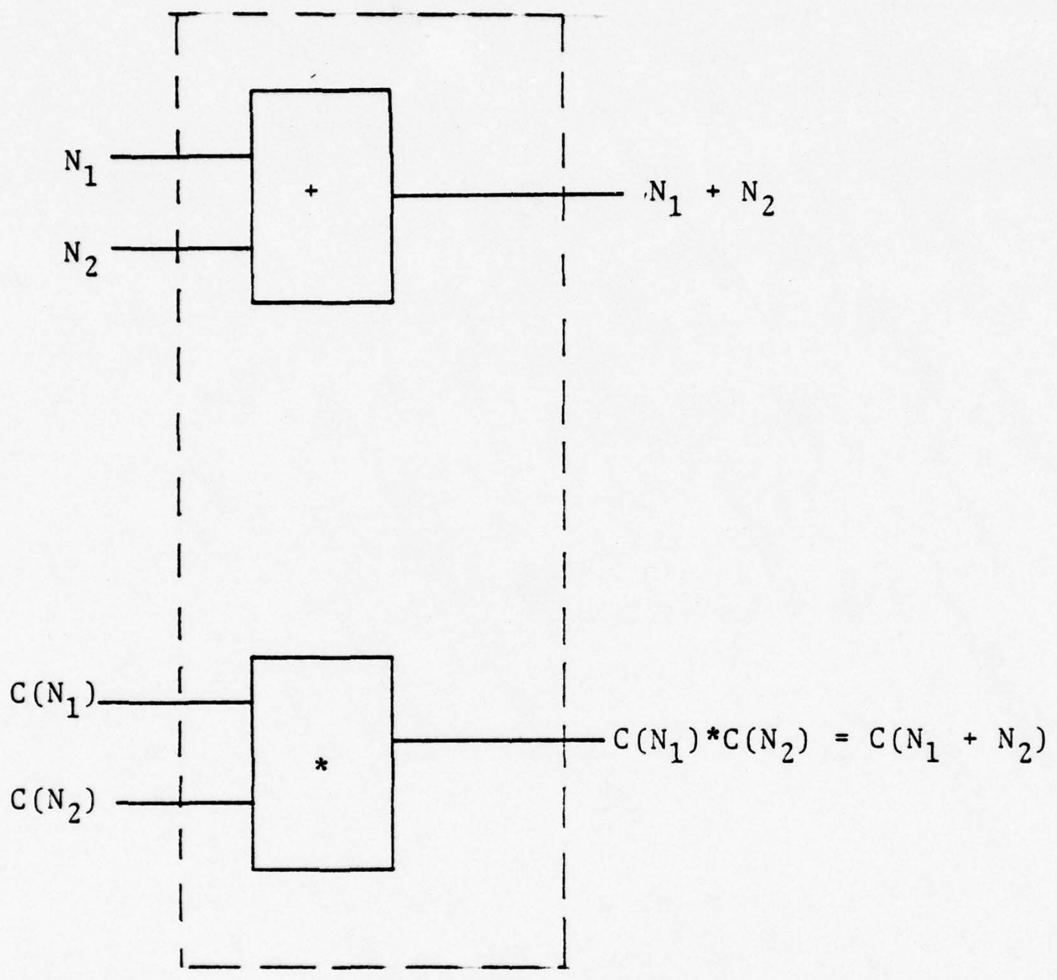


Fig. 5.1. Adder Using Separate Code - Block Diagram.

in a coded form, where b is the number of distinct check symbols, and * is addition modulo b.

Proof

Let S denote the set of all integers N that have the same check symbol as zero; that is, $C(N) = C(0)$. The first step in the proof is to show that S is an ideal in the ring of integers. By definition, zero is in S. Note that $C(0) = C(0 + 0) = C(0)*C(0)$, and therefore, $C(0)*C(0)*\ldots *C(0) = C(0)$, no matter how many factors appear on the left. Thus, if a is in S, then $Na = a + \ldots + a$ is in S, for

$$\begin{aligned} C(Na) &= C(a + a + \ldots + a) = C(a)*C(a)*\ldots*C(a) \\ &= C(0)*C(0)*\ldots*C(0) = C(0). \end{aligned}$$

Also, if a is in S,

$$\begin{aligned} C(-a) &= C(0-a) = C(0)*C(-a) = C(a)*C(-a) \\ &= C(a-a) = C(0) \end{aligned}$$

and therefore -a is in S also. Thus, S is an ideal.

Next, note that $C(N_1) = C(N_2)$ if and only if N_1 and N_2 are in the same residue class of S. This is shown as follows:

$$C(N_2) = C(N_1)*C(N_2-N_1)$$

If N_1 and N_2 are in the same residue class, N_2-N_1 is in S and $C(N_2 - N_1) = C(0)$.

$$C(N_2) = C(N_1)*C(0) = C(N_1+0) = C(N_1)$$

On the other hand, if $C(N_1) = C(N_2)$, then

$$C(N_2-N_1) = C(N_2)*C(-N_1) = C(N_1)*C(-N_1)$$
$$= C(N_1-N_1) = C(0);$$

therefore, N_2-N_1 is in the ideal S, and N_2 and N_1 are in the same residue class of S. Note that this means that if two different integers N_1 and N_2 have the same check symbol, then the ideal S is nontrivial, for then $C(0) = C(N_1-N_2)$, and thus the ideal contains more than just the number zero.

It is know that in a ring of integers any ideal is a principal ideal and thus contains all multiples of some integer b. The residue classes are, therefore, residue classes modulo b. Thus, there is a one-to-one correspondence between residue classes and check symbols. For every j, $0 \leqq j < b$, consider C(j) to be the check integer for the residue class containing j; that is, define $C(j) = j$. Then $i*j = C(i)*C(j) = C(i+j) = (i + 2) \bmod b$, and thus the checking operation is additon mod b.

If there are as many check symbols as numbers, each number has a distinct check symbol, and the checker is equivalent to a duplicate adder. Theorem 4.1 shows that separate codes, defined according to (4.1) are necessarily residue codes. Since $R_B(N_1 x N_z) = R_B(N_1) \times R_B(N_z)$ we can conclude that residue codes have the interesting property of being conserved through multiplication.

B. MULTIRESIDUE CODES

An important generalization of the ideas explored above is found in the theory of multiresidue codes, first introduced by

Dadayev. The multiresidue code in Z_m generated by the positive integers $m_1, m_2 \ldots m_k$ each being a divisor of m is the set of (k+1)-tuples of integers

$$N_c = [N, C_1 (N), C_2(N) \ldots C_k(N)]$$

with N in Z_m and where $C_i(N) = R_{mi} (N)$, $i = 1, 2 \ldots k$. So each code word is formed by one information integer N and k check integers $C_i(N)$. In this study we will assume that each of these integers is represented in its radix-2 form, as indicated for use in digital computers. Since each m_i is a divisor of m it follows that

$$C_i(N_1 + N_2) = R_{mi} (N_1 + N_2) = R_{mi} [C_i(N_1) + C_i (N_2)]$$

So the ith check integer for N_1 N_2 can be calculated as the modulo m_i sum of the ith check integers for N_1 and N_2. As a result, modular arithmetic in Z_m can be performed by k+1 independent adders (or arithmetic units), the first, which we will call the main processor, performing modulo m arithmetic on the information integers, while the other, which we will call checkers, performing modulo m_i arithmetic on the corresponding check integer.

C. AN CODES AND MULTIRESIDUE CODES

Consider $A = \text{LCM} (m_1, m_2 \ldots m_k)$ where LCM (a, b, ... j) denotes the least common multiple of a, b, ... j.

Lemma 4.1

For any N in Z_m the check integers $C_1(N)$, ... $C_k(N)$ in the multiresidue code generated by m_1, m_2, ... m_k uniquely determine and are also uniquely determined by the residue $R_A(N)$.

Proof

$$N = KA + R_A(N)$$

but $A = m_1, m_2, \ldots m_k$ as an immediate consequence

$$R_{m_i}(KA) = 0. \quad \text{So, } R_{m_i}(N) = R_{m_i}[R_A(N)] = C_i(N).$$

Now, suppose $N_1 \neq N_2$ and $R_{m_i}(N_1) = R_{m_i}(N_2)$; then $R_{m_i}(N_1 - N_2) = 0$ $i = 1, 2, \ldots., k$ and $(N_1 - N_2)$ is divisible by m_i for all i. But since $A = \text{LCM}(m_1, m_2, \ldots m_k)$, $N_1 - N_2$ is also divisible by A and then $R_A(N_1) = R_A(N_2)$.

Now suppose that the result of some operation whose correct result is N is in fact

$$N' = [N \oplus F, C_1(N), \ldots, C_k(N)] \tag{5.2}$$

because of a ring error F in the information digit N. It follows from Lemma 4.1 that the assumed correct check integers in N' suffice to determine $R_A(N)$, which can then be subtracted modulo m from the possibly erroneous information integer $N + F$ in N' to give

$$I = [N \ominus R_A(N)] + F$$

or, upon noting that $N \ominus R_A(N)$ is divisible by A and thus of the form AN_1 for some $0 \leq N_1 < B = m/A$,

$$I = AN_1 \oplus F \qquad (5.3)$$

Let's now define for a multiresidue code, its associated AN code as being the AN code with $A = \text{LCD}(m_1, m_2 \ldots m_k)$. Then, equation (4.2) proves that an error F occurred in the addition of two multiresidue coded integers, can be corrected or detected if it can be corrected or detected in its associated AN code. In addition, the converse is also true, as we see, again using Lemma 4.1, from the fact that the same values of the check integers would result from any of the B integers N having the same residue $R_A(N)$ determined by the check integers, so that the AN_1 in (5.3) might have been any of the code points in the associated AN-code. As a consequence, the following important and fundamental result is proved.

Theorem 5.2

If T_c and T_d are any disjoint subsets of Z_m, then a multiresidue code can correct any ring error F in T_c appearing in its information integer, or detect any ring error in T_d appearing in its information integer (assuming the check integers are error-free) if and only if the associated AN-code can correct all ring errors in T_c and detect all ring errors in T_d in its code points. Theorem 5.2 provides the justification to the emphasis given to AN codes, regardless of the fact that separate codes seem to be the most natural choice for most practical applications.

The cyclic AN-codes again assume fundamental importance since in their case, $m = 2^n-1$ and the main processor is then just an ordinary n-bit one's complement arithmetic unit. We note that there is a freedom in choosing a multiresidue code from a given AN-code that can often be used to advantage in simplifying the checkers. The only requirement is that the m_i be selected so that their least common multiple is A. For instance, if A = 315 = 3x5x21, we could select a uniresidue code with $m_1 = 315$. We could also select a biresidue code with $m_i = 5$ and $m_2 = 21$ or a biresidue code with $m_1 = 15 = 2^4 - 1$ and $m_2 = 63 = 2^6 - 1$. This latter choice is the most advisable since the two checkers could then be built as 4-bit and 6-bit one's complement arithmetic units, respectively. This particular value for A generates a cyclic AN-code with length n = 12, so that the main processor would be a 12-bit one's complement arithmetic unit. We will next show that there are useful biresidue codes derived from cyclic AN-codes that have one's complement checkers.

D. MULTIRESIDUE CODES DERIVED FROM CYCLIC AN-CODES

When $m_i = 2^b-1$, not only is the corresponding checker simply a b bit one's complement arithmetic unit as noted previously, but also the residues modulo b are easily formed, i.e., the check integers for N are simple to calculate. (Throughout this section, we shall often use the fact that for any integer $c > 1$, c^i-1 divides c^j-1 if and only if the positive integer i divides the positive integer j.) For, suppose $m_i = 2^b-1$ and the multiresidue code is derived from a cyclic AN-code so that $m = 2^n-1$. Since m_1 divides m, then b divides n, so that the radix-2 form of any N in Z_m may be partitioned into n/b portions each with b bits. If I_i is the integer represented by the i th lowest portion, then

$$N = I_1 + 2^b I_2 + \ldots + 2^{n-b} I_{n/b}$$

and it follows then from (3.3) and (3.4) that

$$R_{m_i}(N) = R_{m_i}(I_1 + I_2 + \ldots + I_{n/b})$$

that is, the check digit $C_i(N)$ is the modulo 2^b-1 sum of the integers I_1, I_2, ..., $I_{n/b}$. Hence, the checker itself (which is a b-bit one's complement arithmetic unit) can form the check digit by adding the n/b integers represented by the consecutive b-bit portions of the radix-2 form of N as this radix-2 form is being read into the arithmetic unit of the main processor.

We now go on to consider some useful multiresidue codes derived from cyclic codes each of whose m_i is of the form 2^b-1 for some b.

Combining the results of Theorem 5.2 and Corollary 4.1, we have immediately the following:

Corollary 5.1

For $m = 2^n-1$ and any even n, the uniresidue code generated by $m_1 = 2^2-1 = 3$ can detect all single modular errors in its information integer.

For these simple codes, the checker is simply a two-bit one's complementarithmetic unit. If n is large, it is certainly reasonable to assume, as we have been doing, that the checker is much more reliable than the main processor.

We now proceed to establish a similar result for single-error-correction which is due to Rao.

Theorem 5.3

For any two integers a and b, $a > 1$ and $b > 1$, the bi-residue code with $m_1 = 2^a-1$ and $m_2 = 2^b-1$ and with $m = 2^n-1$ and $n = \text{LCM}(a,b)$ can correct all single modular errors in its information integer.

Proof

By Theorem 5.2 it suffices to show that $A = \text{LCM}(m_1, m_2)$ generates a cyclic AN-code of length $n = \text{LCM}(a,b)$ having $D_{min} \geq 3$. But A divides 2^i-1 if and only if both $m_1 = 2^a-1$ and $m_2 = 2^b-1$ divide 2^i-1, which in turn occurs if and only if LCM(a,b) divides i. It follows that our choice of n satisfies $n = e(A)$, the exponent of 2 modulo A. Now, by Theorem 3.3, $D_{min} \geq 3$ unless $n = e(A)$ is even and A divides $2^{n/2} + 1$. But if n is even, at least one of a and b must be even, say $a = 2a'$. But a divides n and hence a' divides n/2, which implies that $2^{a'} - 1$ divides $2^{n/2} - 1$ and further implies (for $a' > 1$) that $2^{a'} - 1$ does not divide $2^{n/2} + 1$. Thus, neither can $m_1 = 2^a - 1 = (2^{a'} - 1)(2^{a'} + 1)$ nor $A = \text{LCM}(m_1, m_2)$ divide $2^{n/2} + 1$, so that $D_{min} \geq 3$ for $a' > 1$. The case where $a' = 1$, i.e., where $a = 2$, is trivial.

The single-error-correcting cyclic AN-codes derived from the biresidue codes of Theorem 5.3 are, of course, inferior, as AN-codes, to the perfect codes of Theorem 3.4. On the other hand, the form of the residues m_1 and m_2 makes these codes the practical choice for implementation as biresidue codes. In fact, only uniresidue codes can be derived from the codes of Theorem 3.7 since A is a prime p and the checker would have to do arithmetic in GF(p).

Example 5.1

Choose a = 5 and b = 7 for the biresidue code in Theorem 5.3. Then n = LCM(5, 7) = 35, so that the main processor is a 35-bit one's complement arithmetic unit. The two checkers are 5-bit and 7-bit one's complement arithmetic units, respectively.

One important aspect of multiresidue codes should not be overlooked. The number of code words N in the multiresidue code is always given by m = AB regardless of the number of code words B in the associated AN-code. Particularly for cyclic AN-codes where $m = 2^n - 1$, we see that the number of code words in the multiresidue code depends only on the length n of the cyclic AN-code and not at all on its redundancy $\log_2 A$. A natural definition of the redundancy of the multiresidue code generated by $m_1, m_2, \dots m_k$ is as the quantity

$$r = \sum_{i=1}^{k} \log_2 m_i = \log_2 \left(\prod_{i=1}^{k} m_i \right) \qquad (5.4)$$

Since $A = \mathrm{LCM}(m_1, m_2, \dots M_k)$ it follows that a multiresidue code always has at least as great redundancy as its associated AN-code, i.e., $r \geq \log_2 A$, and that the redundancies are equal if and only if the moduli m_i are pairwise relatively prime. We note also that r, as given by (5.4), is an approximate measure of the number of bit positions required for the radix-2 forms of the check integers in a multiresidue code.

A final comment on Theorem 5.3 should also be made. Suppose a divides b so that $n = b$ and hence $A = m = 2^n - 1$. The associated cyclic AN-code has only $B = 1$ code point and hence $D_{min} = \infty$, so that it can correct all errors in its code points. By Theorem 5.2, the multiresidue code can correct all errors in its information integer if the check integers are error-free. This is not as surprising as it may seem when one observes that $m_2 = m$, so that the second checker is in fact a duplicate of the main processor! What this does point out is that our working assumption that the checkers are much more reliable than the main processor is certainly unacceptable when some m_i coincides with m.

E. SYNDROMES IN MULTIRESIDUE CODES

Suppose that

$$V = (I, J_1, J_2, \ldots, J_k) \qquad (5.5)$$

is a (k+1)-tuple with I in Z_m and J_i in Z_{m_i} that is a possibly erroneous code word in a multiresidue code. We define the syndrome S(V) of V in the multiresidue code to be the k-tuple

$$S(V) = [R_{m_1}(I-J_1), \ldots, R_{m_k}(I-J_k)] \qquad (5.6)$$

Notice that if V is an actual code word N, then $J_i = R_{m_1}(I)$, so that from (3.2) we have

$$S(V) = [0, \ldots, 0] \qquad (5.7)$$

and conversely, (5.7) holds only if V is a code word.

Notice that that conclusion does not require our hypothesis that error can occur only in the information integer.

Now, if errors can occur only in the information integer so that V has the form N' given in (5.2), we see that again with the aid of (3.2) that (5.6) gives

$$S(N') = [R_{m_1}(F), R_{m_2}(F), \ldots, R_{m_k}(F)] \qquad (5.8)$$

But (5.8) together with Lemma 5.1 shows that S(N') uniquely determine, and in turn is uniquely determined by, $R_A(F)$, which, we recall from (3.5), is the ordinary syndrome S(A + F) for the result A + F in the associated AN-code. We have proved the following:

Lemma 5.2

The multiresidue code syndrome S(N') for a result N' = $[N + F, C_1(N), \ldots, C_k(N)]$ *uniquely determines*, and in turn is uniquely determined by the syndrome S(A + F) of the associated AN-code.

Now, combining the results of Theorem 5.1, Lemma 5.2, and Theorem 3.3, we obtain the following:

Theorem 5.4

Let T_c and T_d be any disjoint subsets of Z_m. Then, a multiresidue code can correct all ring errors in T_c in its information integer, and can detect all ring errors in T_d in its information integer (assuming the check integers are error-free), if and only if there is no error F_1 in T_c that gives the same syndrome (5.8) as some other error F_2 in either T_c or T_d.

Note that, according to (5.5), the checkers themselves can be used to calculate the syndrome S(V) = S(N'), since

$$R_{m_k}(I-J_i) = R_{m_k}(I) \ominus J_i \text{ for } J_k \in Z_{m_k} \quad (5.9)$$

This syndrome may then be used in an error correction and detection procedure for multiresidue codes entirely analogous to that suggested for AN-codes.

VI. IMPLEMENTATION OF ARITHMETIC CODES

We will in this chapter address briefly the question of implementation of arithmetic codes. The real problem in implementation is the decoder, that is, the device that checks the results for correctness, and that does the actual correction if a correctable error is found to have occurred.

It is at the point interesting to note that the design of a decoder will depend upon the user choice of the sets T_c and T_d, respectively the set of correctable and detectable errors. It is to be noticed that Theorems 3.2 and 3.6 do not determine 5 and s.

For instance, for a code with minimum distance $D_{min} = 6$, decoders can be specified that will:

a. Correct all single or double errors and detect all triple errors ($t = 2$, $s = 1$).

b. Correct all single errors and detect all double, triple or quadruple errors ($t = 1$, $s = 3$), or yet

c. Detect all errors with weight less than 6 ($t = 0$, $s = 5$). All these choices give $2t + s < D_{min}$.

Figure 6.1 a, b and c illustrates respectively the decoders characterized in a, b and c above.

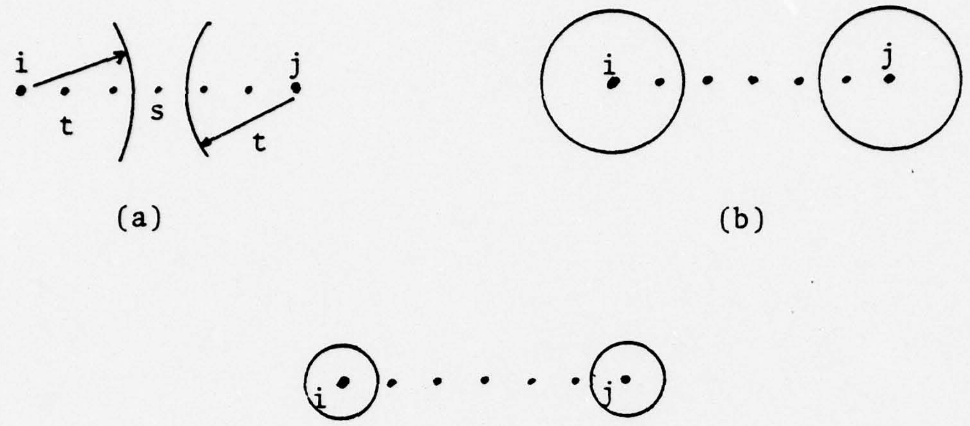


Fig. 6.1. Decoding Rules for $D_{min} = 6$
Any actual result in sphere S_i is translated into code point i. Points not included in any S_i are detected as errors.

A. DECODER FOR AN AN-CODE

The decoding algorithms for an AN-code has been briefly outlined in Chapter IV, and is next discussed in further detail. If the actual—possibly erroneous—result of an operation is Y, the first step in the decodification process if to calculate the syndrome S(Y). A syndrome S(Y) = 0 is interpreted as a correct result.

If a nonzero syndrome results, the next step is to look up in a table the correction C = - F, corresponding to S(Y).

Theorem 3.3 guarantees that there is a one to one correspondence between the syndrome of any correctable error and the error itself. The correct result is therefore

$$I = Y \oplus C$$

Figure 6.2 presents a flow chart of the above described algorithm. Notice that steps 4, 5 and 7 are not required if only detection of errors is desired.

We see immediately that only two of the steps in the algorithm involve some difficulty.

a. The determination of the syndrome $S(Y) = R_A(Y)$, since this calculation requires the determination of the remainder of the division of Y by A, and it does not seem very smart to check an addition with an algorithm that requires a more involved operation such as division. This problem will be addressed in **Section** C.

b. The lock up operation to find the correction, given the syndrome. This operation, even though conceptually trivial may require a very extensive table, when large or even moderate distance codes are used and if a decoder with large error correcting capability is desired. Notice however, that this step is unnecessary for decoders that only make error detection, and further, that the size of the table is considerably reduced if only a small set of errors is to be corrected, as in the case of single error correction.

The look up operation also may result in considerable delays, but use of associative memories may be a way of eliminating such excessive delays.

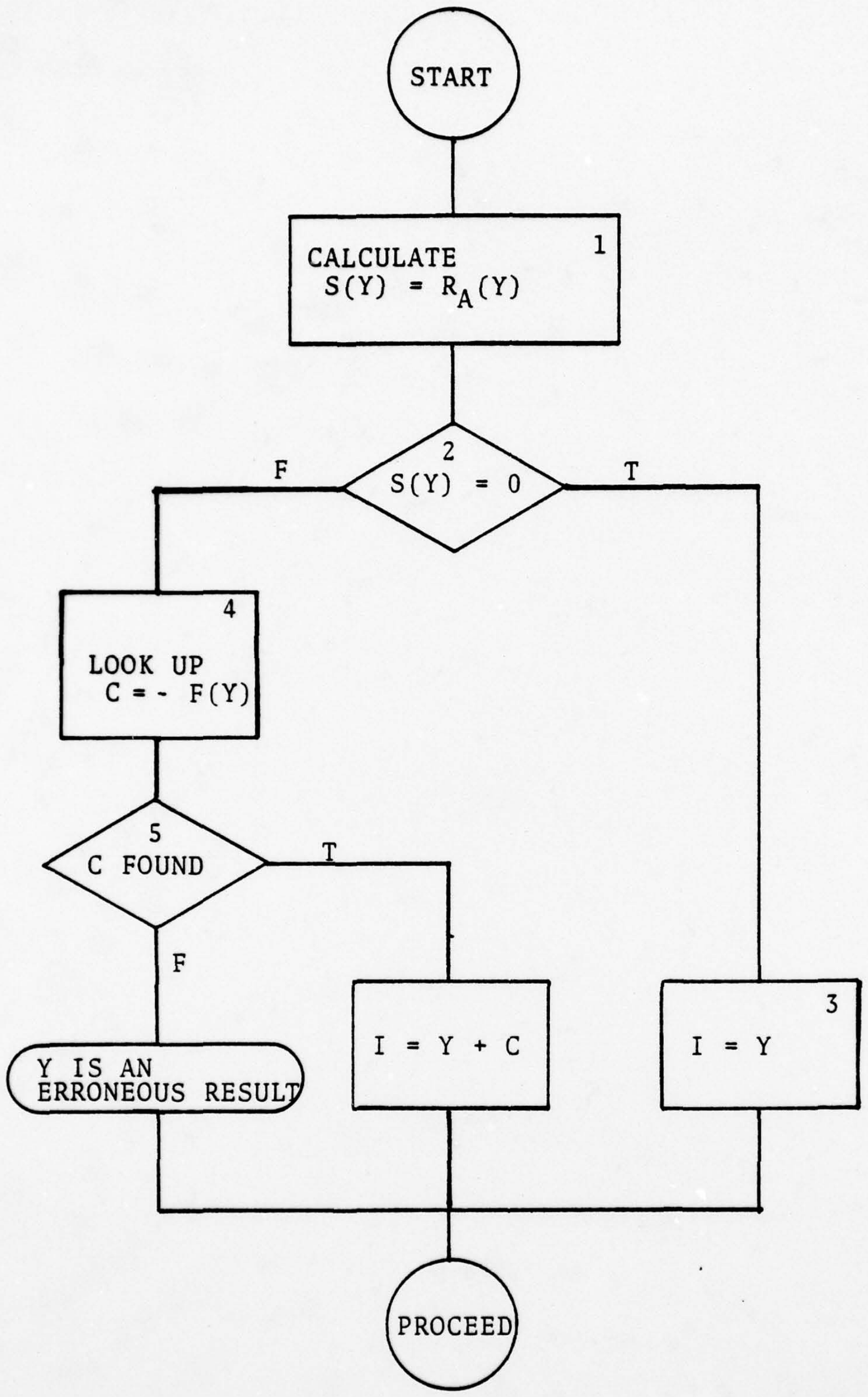


Fig. 6.2. Flow chart for decoding algorithm for AN-codes.

B. DECODER FOR RESIDUE CODES

The algorithm that decodes a multiresidue code is very similar to that described in Section A.

If N' is a possibly erroneous result of an addition, then the syndrome S(N') is, according to (5.8)

$$S(N') = [R_{M_1}(F), R_{M_2}(F) \ldots R_{M_k}(F)]$$

As pointed out in (5.9) the syndrome can be evaluated by the checkers themselves, provided $R_{M_i}(I)$ is known. So, again, the determination of S(N') implies the determination of the remainder of a division of I by M_i. Lemma 5.2 assures the possibility of constructing a lock-up table that translates S(N') into the correction to be applied to N' to produce the correct result.

C. DETERMINATION OF $R_M(N)$

Let's now consider the problem of determining the residue of an arbitrary integer N, represented in its radix-2 form, modulo another integer M.

As we have shown previously, this is an essential step in any algorithm intended to decode an arithmetic code.

We will show that this determination can be accomplished by means of a modulo M multiply by 2 shift register, with block diagram is represented in Fig. 6.3.

Fig. 6.3 gives a false impression about how complicated the circuit is, because it attempts to make a general, easy to understand schematic representation of the device.

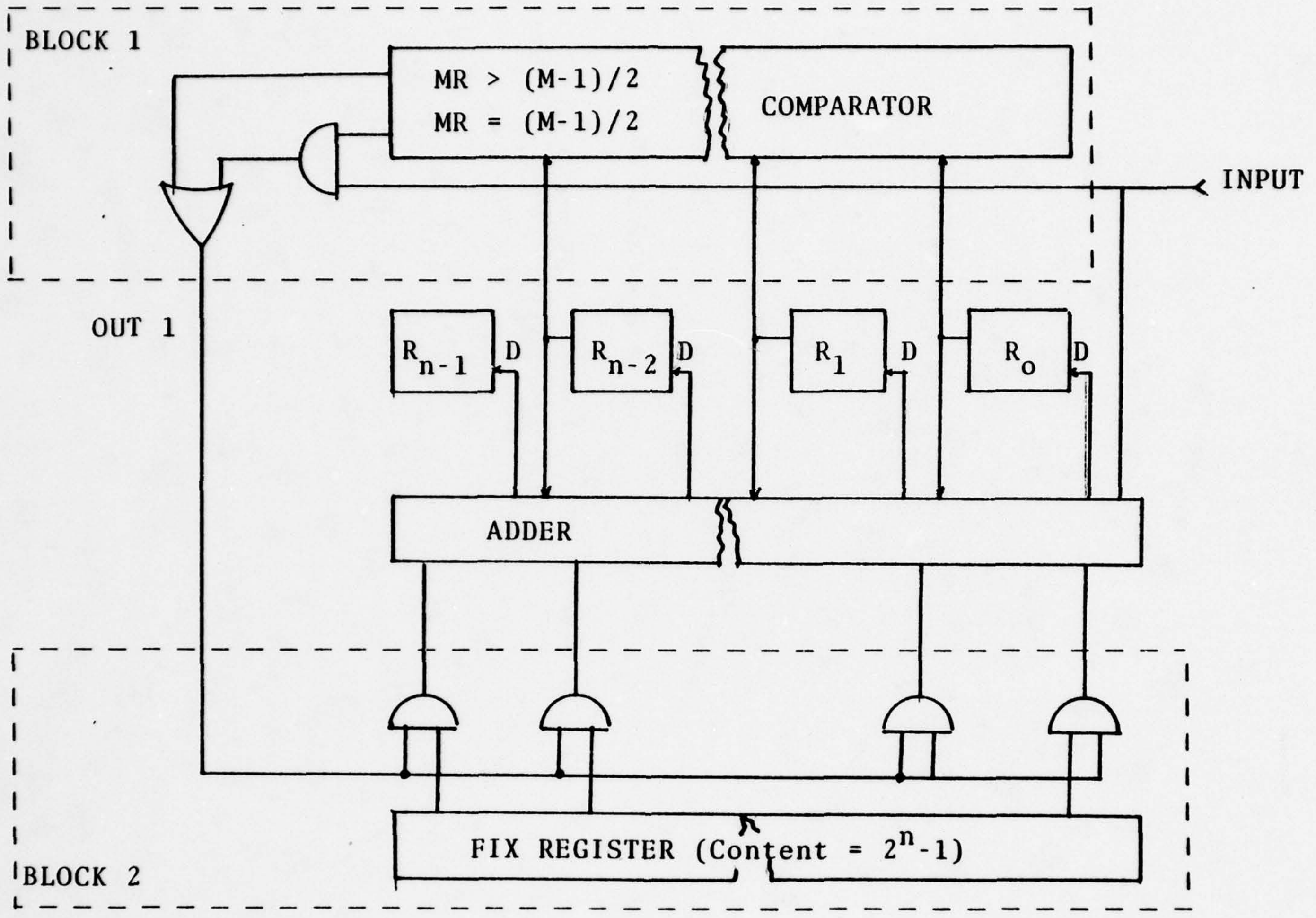


Fig. 6.3. Modulo M, multiply by 2 shift register.

Actually BLOCK 1 is just a (n+1) input, one output combinational logic network, and BLOCK 2 is just a few pieces of wire, correcting the output of BLOCK 1 (OUT 1) to the proper adder inputs to form the negative of M, modulo 2^n, in such a way that the adder actually subtracts M from the content of the register whenever OUT 1 = 1. The operation of the device is self explanatory.

N is shifted into the register, most significant binary digit first. Whenever the contents of the register is greater than M-1/2, or is equal to M-1/2 but the input is 1, the content of the register after next shift would be greater than M, but less than 2M. If M is subtracted at shift time, then the content of the register will be conserved less than M, and $R_M(N)$ will be stored in the register after N is completely shifted in.

An example of such a circuit is given in Figure 6.4 which represents a modulo 5, multiply by 2 shift register, that will calculate $R_5(N)$, regardless the length of the radix-2 form of N. To illustrate the operation of the circuit in Figure 6.4, let's take $N = 10110 = 22_{10}$. Table IV gives the successive states of the register as N is shifted in.

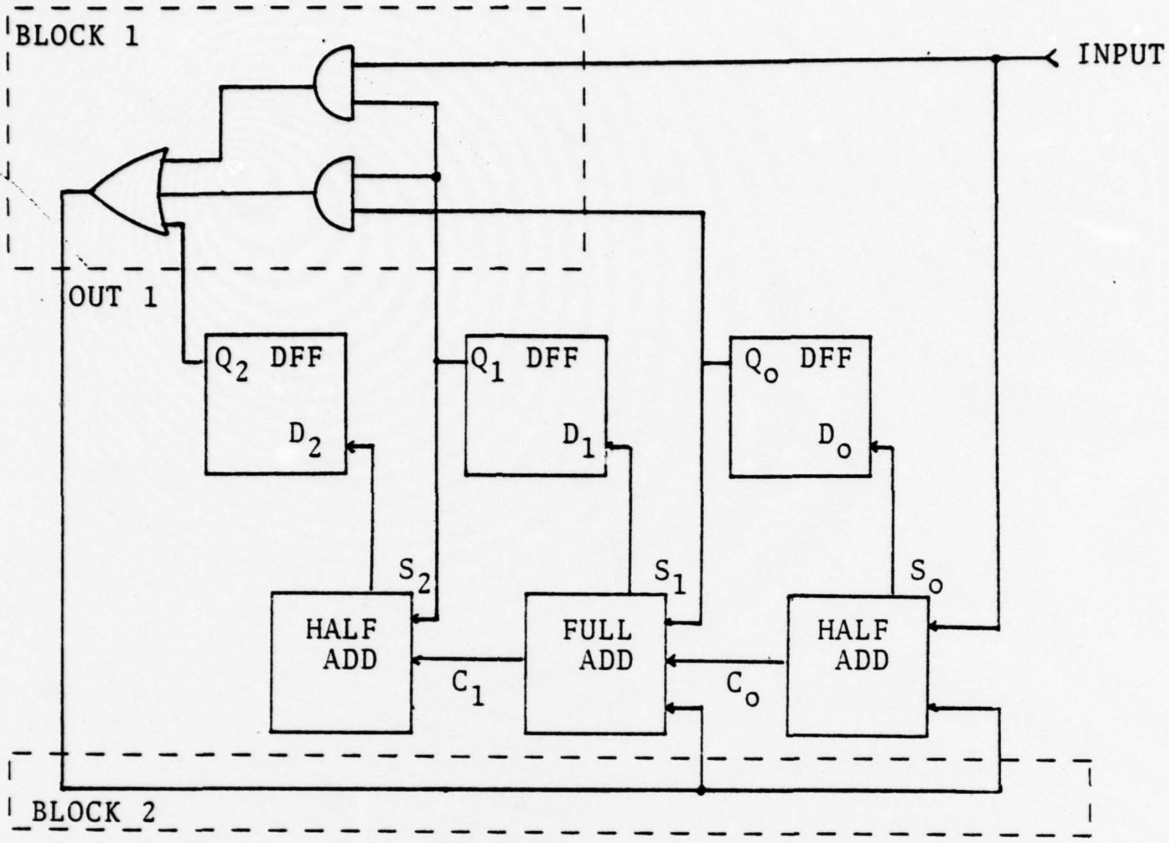


Fig. 6.4. Modulo 5 multiply by 2 shift register.

TABLE IV

Clock Pulse	Register	Input
0	000	1
1	001	0
2	010	1
3	000	1
4	001	0
5	010	x
	$=R_5(22) = 2$	

Operation of Modulo 5 Divided by 2 Shift Register

VII. CONCLUSION

This thesis constitutes a tutorial study on arithmetic codes.

It seems appropriate at this point to emphasize the fact that the problem of designing an arithmetic code is essentially that of finding a generator A which generates an AN-code with the desired error correcting and detecting capabilities for addition in the ring of integers modulo B. This is evident for AN-codes. For residue codes it translates into finding an integer $A = \text{LCM}(m_1, m_2, \ldots, m_k)$ which generates an AN-code which is capable of correcting all errors in T_c, and of detecting all errors in T_d, *respectively the sets of* correctable and detectable errors in the information integers of the residue code defined by $m_1, m_2, \ldots, m_k$ and the number of code points.

The theory discussed in the previous chapters provides some guidance and some tools to solve that problem. We have mentioned in Chapter 4, that perfect, single error correcting AN-codes, analogous to Hamming parity check error correcting codes are known. Also, large distance cyclic AN-codes have been discovered by Mandelbaum and Barrows. Unfortunately, moderate distance, systematically constructed AN-codes are not known. A proof of a BCH-like bound for AN-codes such as the Hartman conjecture outlined in Chapter IV would constitute a major theoretical contribution to the field of arithmetic codes.

AD-A067 579 NAVAL POSTGRADUATE SCHOOL MONTEREY CALIF F/G 9/2
ARITHMETIC CODES.(U)
DEC 78 T J PEREIRA

UNCLASSIFIED NL

2 OF 2
AD
A067579

END
DATE
FILMED
6 -- 79
DDC

On the practical side of the matter we recognize that extensive circuitry is required to decode arithmetic codes. The search for more efficient algorithms, applicable to particular classes of codes, that could be easily implemented in hardware is an attractive and promising area for future research. The analogies between cyclic AN-codes and cyclic parity check error correcting codes may suggest the possibility of decoding with shift registers.

We have through this paper made the assumption of perfect decoding, since our main concern was the structure of arithmetic codes. In real life, this assumption is quite unrealistic. The evaluation of the improvement in reliability that can be achieved by using arithmetic codes under imperfect decoding is also an interesting subject for additional effort.

Finally, it is appropriate to mention that this thesis is not intended to present original contributions. The material discussed here has been discussed by many authors, such as Peterson, Massey, Garcia, Hartman, Avizienis and many others.

BIBLIOGRAPHY

1. Avizienis, A., Concurrent Diagroup of Arithmetic Processors, Digest First Annual IEEE Computer Conference, pp. 34-37, September 1970.

2. Mandelbaum, Arithmetic Codes with Large Distance, IEEE Transactions Information Theory, IT-13, pp. 237-242, April 1967.

3. Avizienis, A., Design of Fault Tolerant Computers, AFIPS Conference Proceedings, p. 31, 1967.

4. Avizienis, A., Fault Tolerant Systems, IEEE Transactions on Computers, C-25, NPR 12, pp. 1304-1311, December 1976.

5. Brown, D. T., Error Detecting and Error Correcting Binary Codes for Arithmetic Operations, IRE Transactions on Electronic Computers, EC-9, pp. 333-337, 1960.

6. Garner, H. L., Error Codes for Arithmetic Operations, IEEE Transactions on Electronic Computers, EC-15, pp. 763-770, October 1966.

7. Massey, J. L., and Garcia, O. N., Error Correcting Codes in Computer Arithmetic, Advances in Information Systems Science, Vol. 4, Plenum Press, N.Y., London, 1972.

8. Barrows, J. T., A New Method for Constructing Multiple Error Linear Residue Codes, Report R-277, Coordinate Science Lab, University of Illinois, Urbana, January 1966.

9. Garcia, O. N., Error Codes for Arithmetic and Logical Operations, PhD. Thesis, Report D-69-03 University of Maryland, College Park, 1969.

10. Chien, R. T., Hong, S. J., and Preparata, F. P., Some Contributions to the Theory of Arithmetic Codes, Proceedings of Hawaii International Conference on Systems Sciences, pp. 460-462, 1968.

11. Chien, R. T., Hong, S. J., and Preparata, F. P., Some Results in the Theory of Arithmetic Codes, Coordinated Science Laboratories, University of Illinois at Urbana, Report #R-440, October 1969.

12. Chang, S. H., and Wu, N. Tsao, Discussion on Arithmetic Codes with Large Distance, IEEE Transactions, Information Theory, IT-14, pp. 174-176, January 1968.

13. Dadayev, Y. G., Arithmetic Divisible Codes with Correction for Independent Errors, Engineering Cybernetics, pp. 79-88, November 1965.

14. Hartman, W. F., Arithmetic Codes, Ph.D. Dissertation, University of Notre Dame, Indiana, May 1975.

15. Peterson, W. W., Error Correcting Codes, MIT Press, Cambridge, Massachusetts and London, 1961.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 62 Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Associate Professor Mitchell L. Cotton Department of Electrical Engineering Code 62Cc Naval Postgraduate School Monterey, California 93940	1
5.	Commander Tarcisio J. C. Pereira Av Vieira Souto 298-701 Ipanema Rio de Janeiro, BRAZIL CEP 22420	1